

Base de Dados

Grupo PI 810

- 66 : 14 24 708

Márv. poss: ① meados 2009

Descrição lógica dos dados na BD → chama-se de esquema (schema)

### 1º Gênero de bases de dados

modelo Heterogêneo → os dados são armazenados em ónus, um registo poi ~~ao~~ é associado a N registros

1:N, o operação de proxy contrária fóra com que se tenta de permanecer a ónus toda, também terá a replicação de inf. em casos de o respeito ~~with~~ não estiver associado o dif. nomes.

Este modelo é útil em sistemas de associação sequencial.

~~no entanto não~~

mas para realizar pesquisa intensa traz grandes desvantagens pois teria de haver uma ónus intensa, daí replicação de dados.

### Modelo de Rede

O mesmo registo está associado a vários nós e ~~ao~~ a partir de um é possível alcançar outros, não é necessário permanecer todo o base de dados para a encontro.

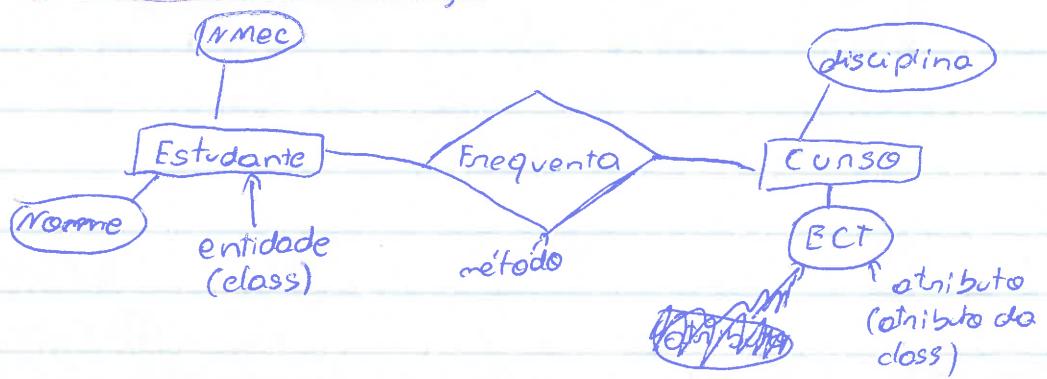
## Desenho Base de Dados

- Análise de Requisito
- Model E/R → diagrama ilustrativo das funções
- Esquema lógico
- 
- 

Análise de Requisitos — levantamento detalhado de toda a inf. associada ao problema.

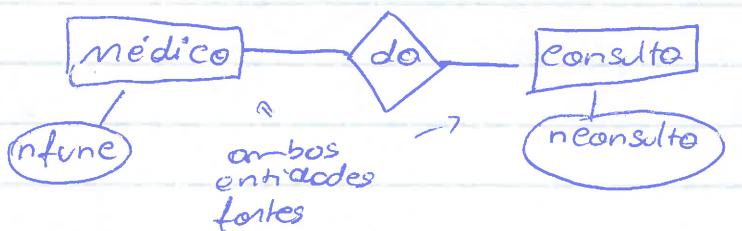
- Observação
- Compreender o problema
- Filtrar os dados
- Discussão de aspectos dúbios, folhas

## Modelo Entidade/Relação



• Entidades Fóntes → têm um atributo que os identifica

- " " Fóntes → Para identificá-la é necessário associá-la a outra classe



Num cenário em que o nº de consulta é feito por médico, por exemplo: cada médico tem a consulta 1, 2, 3, ... assim só hó retípico de inf. e a única maneira de identificar uma consulta, é associá-la a um médico, o médico é uma entidade forte e as consultas são entidades fracas

## Relações



classificação de relações

- Grav - nº de entidades
- Obrigatoriedade - uma entidade pode ou não ser obrigada a participar numa relação

## Relação

relações → recursivas:

- assimetria e intensidade

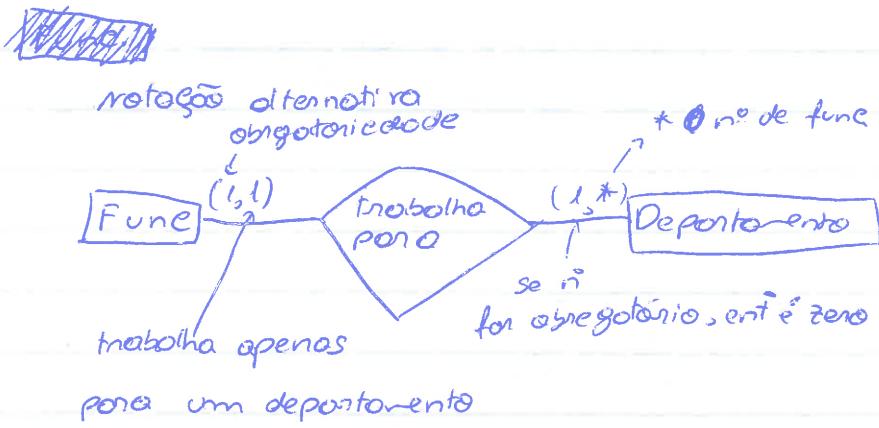


- Um médico tem N consultas mas as consultas apenas têm um médico

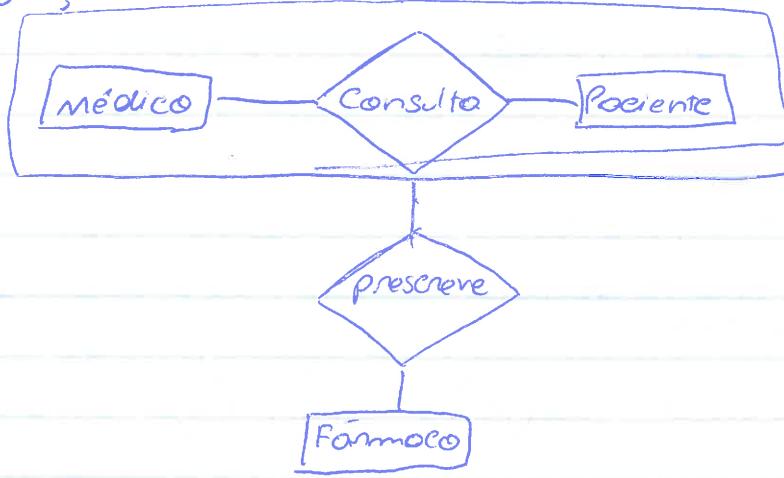
## Notação de Chen



um funcionário gene um departamento



## Agregação



## 1º conjunto de slides

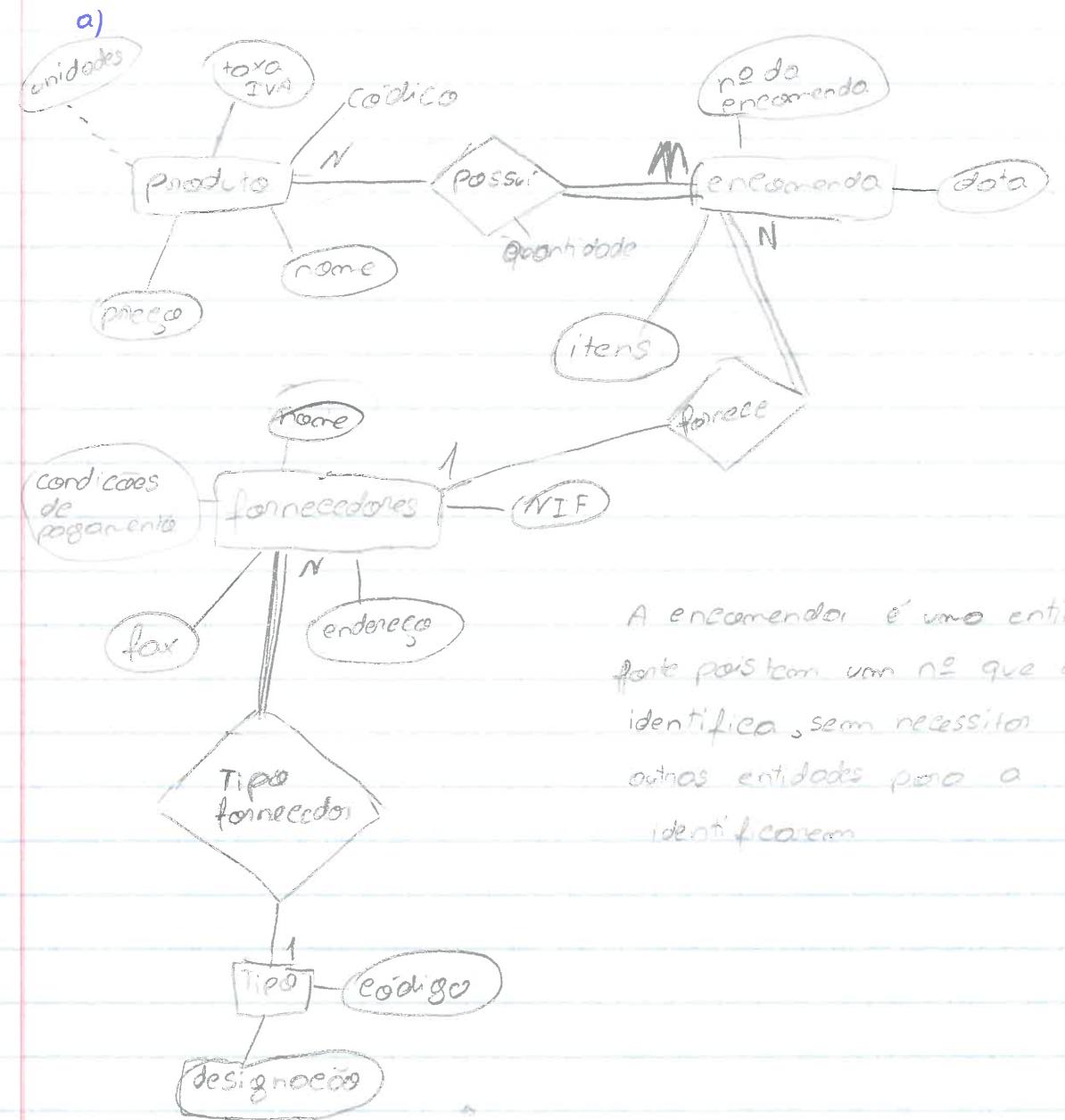
slide 28

A opção A é mais flexível, um empréstimo pode estar associado a M clientes, enquanto que a B apenas permite a associação a um cliente.



## Fazer a modelação da empresa

## Problema 2.-1

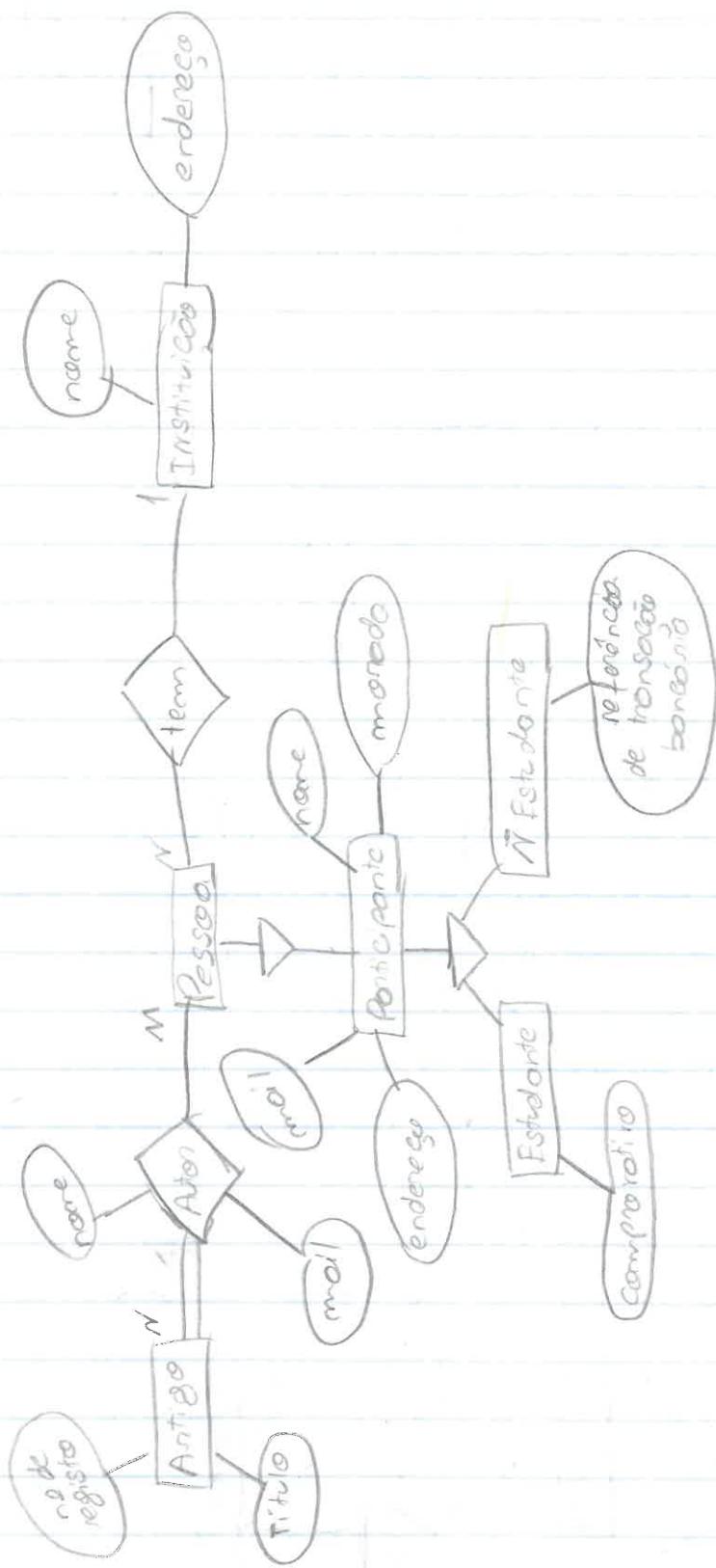


A encenadoria é uma entidade forte pois tem um nº que a identifica, sem necessitar de outras entidades para a identificarem.

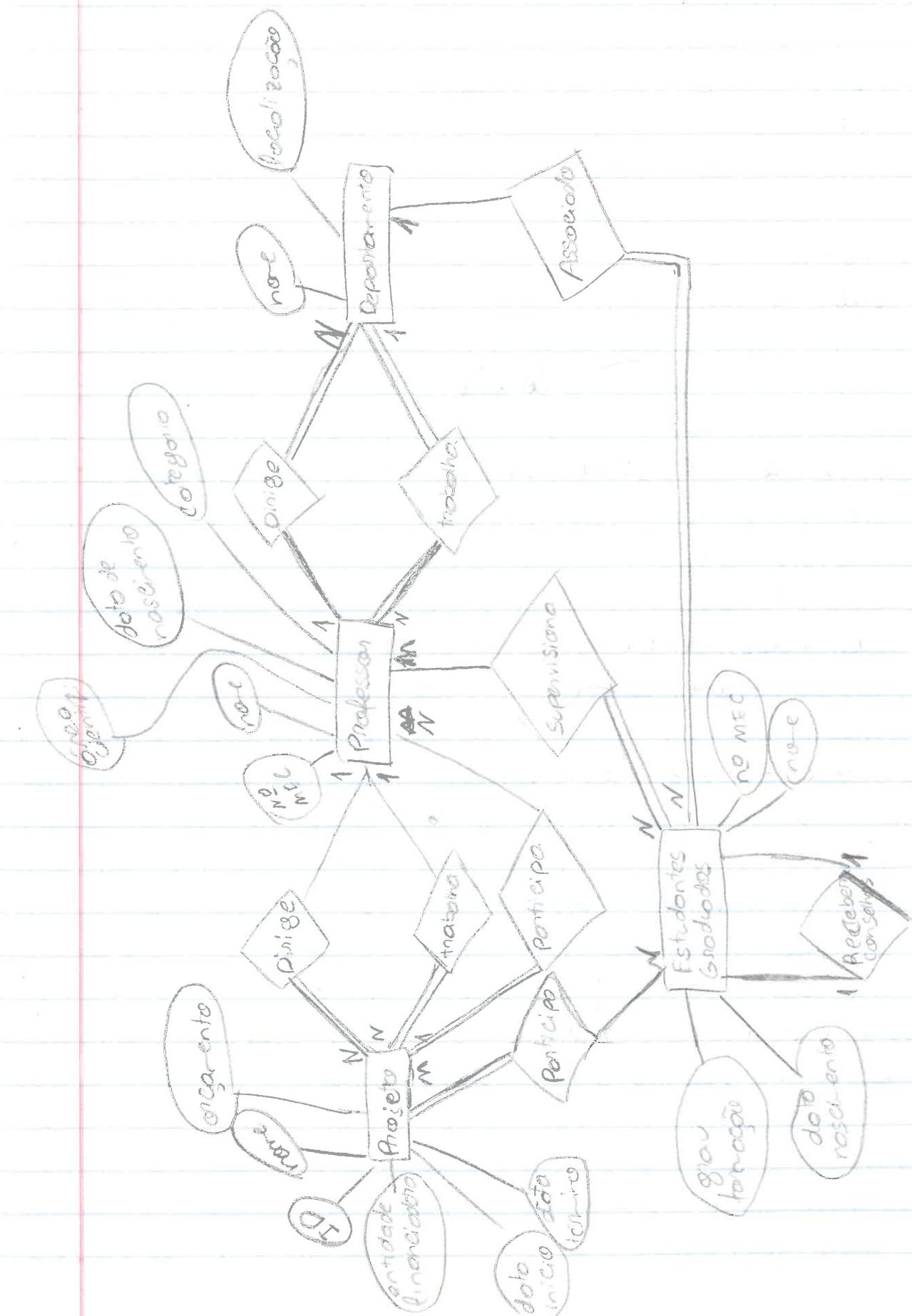
## Problema 2.2

Entidades : Attributos :

Problema 2.34



Problema 2.4



## Modelo Relacional

Todos os bases de dados usam um modelo relacional, este tendendo tornar-se popular nos 90

Recomenda-se primeiro aprender modelo relacional e de seguida Information Retreived.

O modelo relacional baseia-se na teoria dos conjuntos  
usa a noção "relações"

estrutura de  
dados onde n̄ há  
repetição de dados

Criamos uma matriz (Tabelas) em que cada instância  
é um registo

Esquema da relação → Nome do esquema com a lista de Atributos  
Ex: pessoal(nome, bi, idade)

Relação chaves → uma ~~relação~~ de identificação ou relações  
elementos

Chaves → elementos que identifique de forma única a instância  
da tabela

Sabemos que o conjunto dos atributos identificam de forma  
única um elemento pois sabemos que n̄ há repetições

Há sempre uma superchave, que é o conjunto dos atributos.

## Exemplos:

Estudante (Nome, mail, NMFC, curso)

## Superchaves:

Nome, Email, NMFC

Nome, NMFC

:

NMFC

Email

NMFC → subconjuntos

Email → que n̄ pertencem

a cardinalidade

de superchave

↓

chave candidata  
a primária

Paremos por E → A escolha da chave primária depende de  
até onde a alguns fatores como o facto de futuramente mudar,  
fatores → o email pode mudar, uma pessoa pode mudar  
o email.

Assim o NMFC torna-se a uma chave primária e o  
email, uma chave única

chaves estrangeiras → chaves primárias de tabelas que são  
usadas noutras tabelas, que n̄ são a tabela em que  
é primária, para identificar algum ele.

## Restrição de Integridade → Tópico muito importante

- Domínio
- Entidade

umas das restrições no  
domínio que se pode efectuar  
é declarar obrigatoriedade

Ex um trabalhador tem que trabalhar  
no departamento, então obriga o atributo V  
de Departamento, em que trabalho n̄ pode estar null

## Regras Codd

Verificam se realmente é possível encher de selecionar uma base dados.

Ex 12

4 - Os metadados (informações de como são divididos e organizados os dados) devem ser armazenados da mesma forma que outros dados são.

Passo do modelo Entidade para model. relacional usou as entidades fontes para criar tabelas

Pelas entidades fontes usa-se o nome privativo das entidades dominantes como nome estrangeiro

Nos casos 1:1 sem nenhum participante obrigatório, também se cria uma tabela para o selecionamento.

P3.1

Cliente

Nome	Endereço	num. conta	NIF
------	----------	------------	-----

Aluguer

Número	Duração	Data	cli_NIF	Boleto num	V.E.I. num
--------	---------	------	---------	------------	------------

Boléto

Nome	Número	Endereço
------	--------	----------

Veículo

Nome	matrícula	lotação	Tipo_Veí_Cod
------	-----------	---------	--------------

Tipo\_Veículo

Designação	oncondicionado	Código
------------	----------------	--------

Ligeiro

Tipo_Veí_Cod	num_lugares	Pontos	Combustível
--------------	-------------	--------	-------------

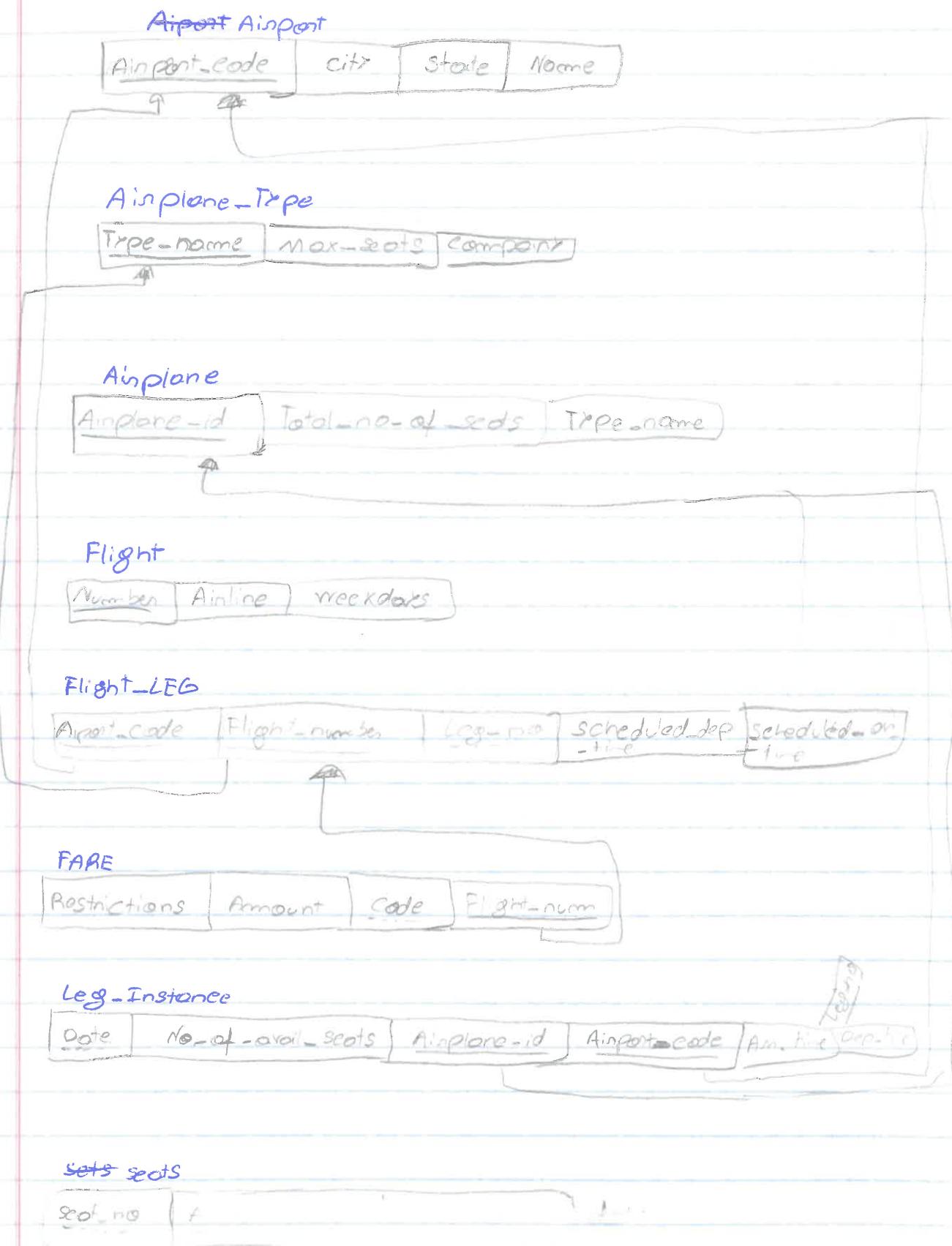
Pesados

Tipo_Veí_Cod	Peso	Passageiros
--------------	------	-------------

Similitude

vi_cod	ve_cod
--------	--------

P3.2



Ponto UX

18:00 | 18:20

U - REC

21:00 | 06:00+1

REC - RS

09:00 | 14:00

P2.2

medico

especialidade

nº identificação

name

prescrição

nº presc.

nº utente

data

Formação

data Proc

paciente

nº utente

name

data nasc.

endereço

Fórmoco

nº registo rec.

nº presc.

Formula

name fórmula

name

farmacia

nº registo rec.

nº presc.

Formula

name farmacêutico

name

farmaceutica

endereço

name

nº registo rec.

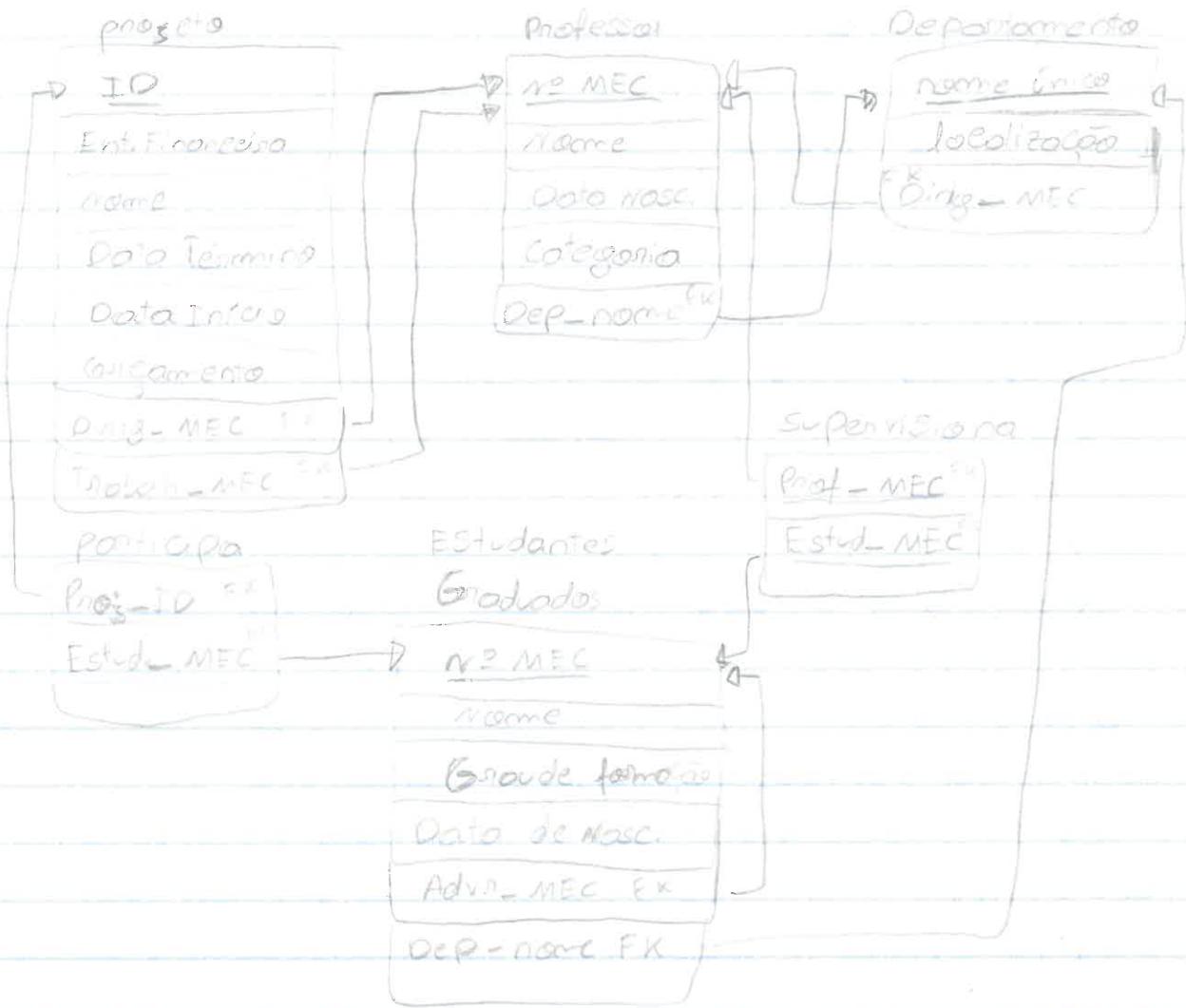
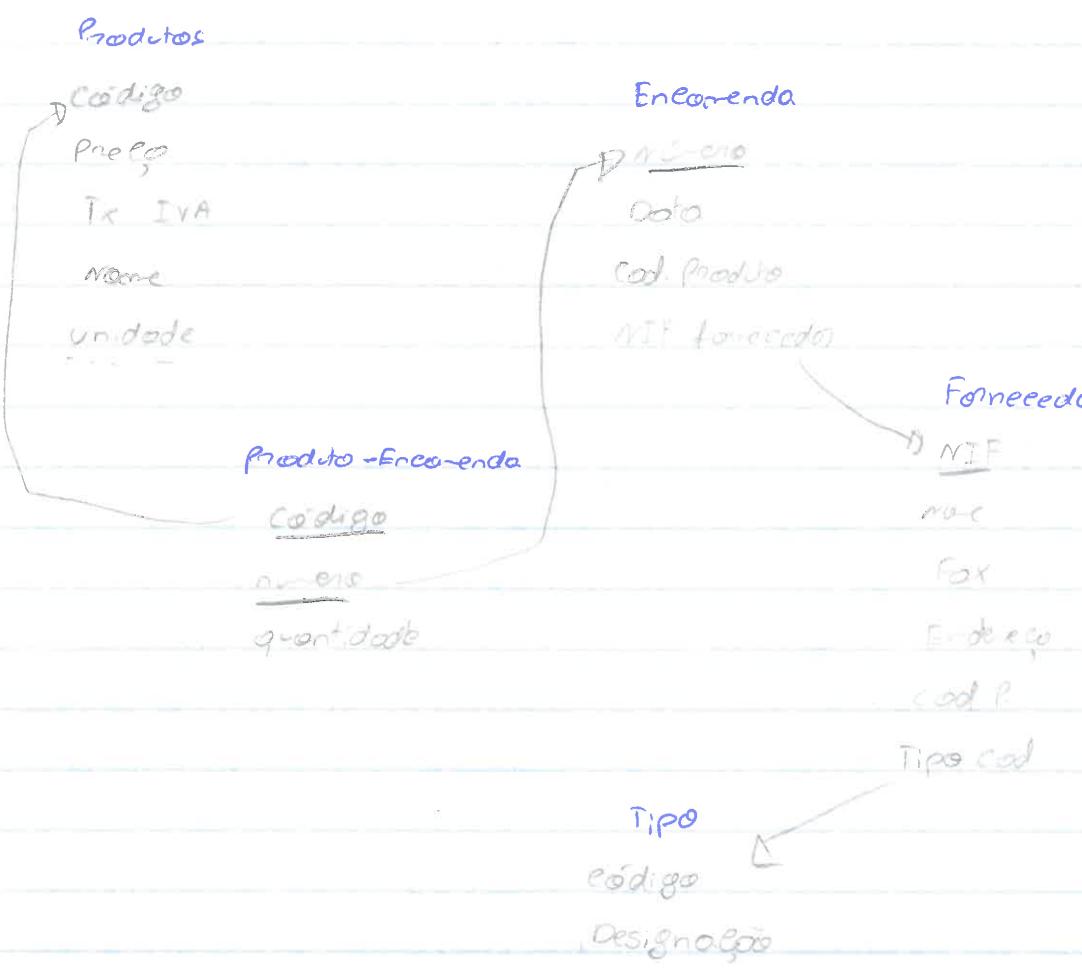
contém

nº presc.

name fórmaco

nº Registo rec.

P2.1



Como passar o modelo Relacional para SQL?

- usando ~~o sql server~~ portugues

Management Studio → editor de texto para SQL

Linguagem O SQL

2 sublinguagens principais:

DDL → Data Definition Language

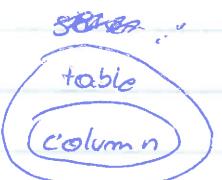
DML → Data manipulation language

1 sublinguagem de BD

↪ Controlo de base de dados

## Objetos SQL

Há uma hierarquia entre objetos



Cada instrução termina com ";"

Comentário → " -- "

Comentário bloco /\* ... \*/

Data Definition Language → Permite definir vários entidades do DB

## Nota importante:

Sempre que se usa um SGBD deve-se consultar a sua documentação.

Criar Base de Dados:

CREATE DATABASE Compan;

DROP " "

CREATE SCHEMA

Tipos de dados

Deveremos utilizar tipos de dados

• Numbers

standard para aumentar a

• Date and Time

possibilidade do sistema

• Characters and Strings

Objetos como imagens e outros ficheiros, não devem ser colocados na Base de Dados em si, cria vários problemas de backup.

Ex: Matrícula sabemos sempre o  
ultimo dígito

char(n) → cadeia de caracteres com tamanho fixo n

varchar(n) → " " " com tamanho máximo n

Ex: Nome de uma pessoa

~~(não é)~~  
~~→~~ ~~preciso~~

numeric(p,s)

ou

decimal(p,s)

Ex: numeric(2,4) → 000012  
p      s      ponto decimal

## Domínio

Permite-nos definir tipos de dados, aplicando restrições

Ex: CREATE DOMAIN

comsalario

NOT NULL CHECK (comsalario > 457)

SQL Server

Há o type → apenas permite verificar se é ou não null

CREATE TABLE EMPLOYEE (Nome varchar(15), SSN INT

atributo      domínio

Ainda podemos aplicar restrições aos atributos  
à extrema após a separação por vírgulas.

Tal como em Java tem haver consistência no escrito, Ex: ~~As classes~~  
~~As classes~~ 6 nome das classes só em letra maiúscula.

UNIQUE → O atributo não é repetido mas pode ser null

Uma das maneiras de garantir que representar um relacionamento obrigatório é adicionar NOT NULL

Se Elemosmos a Entidade Exja chave primária é estrangeira numa entidade e não é incluída pois não viola a integridade da Base de Dados.

Assim podemos preparar as entidades em caso dessas operações

"On delete" → podemos definir que operações ocorrem juntas em caso desta operação.

## Algebra Relacional

Seleção → ato de selecionar uma linha usando uma condição booleana.

Denotada por Q (sigma)

Ex:

$Q \text{ Dno} = 3$  (emplorc)

↪ trabalhadores do depo. 3

$Q \text{ Dno} = 3 \text{ AND } \underline{\text{salario}} > 1400$  (emplorc)

traba. no depo. 3 com salário superior a 1400

Em SQL:

~~SELECT \* FROM EMPLOREC~~  
WHERE Dno = 3

Projeção → selecionar colunas

→  $\Pi_{Lname, Fname} (Q \text{ Dno} = 3)$

Das linhas selecionadas apenas projetar a Lname e o Fname

SQL:

$\text{SELECT DISTINCT } \begin{matrix} \text{LName, Fname} \\ \text{WHERE ...} \end{matrix}$

O distinct serve para mostrar os valores repetidos

Também possível renomear

União - A união de 2 tabelas requer que as tabelas tenham o mesmo tipo de atributos bem como o mesmo nome

$\cup$   duplicados são eliminados  
 $\text{UNION (ALL)}$  → não elimina dupl.

Intersetção  $\cap$  

Diferença



## Produto cartesiano

Todos os combinações possíveis entre dois conjuntos

Ten oténego: pode originar inválidos ~~e~~ entradas na tabela

Outra coisa - os tuplos que não são emparelhados também são  
selecionados, mas como não emparelha com o da outra tabela  
na tabela em não emparelham mete a null

Pode ser útil para saber quais os tuplos que não participam  
na emparelhagem

## Agregações

ssn	salary	Dno
	900	1
	700	2
	1400	1
	1300	3
	2000	3
	500	3

Faz a média de cada

Departamento

Dno  $\{$  avg(salary)

Dno	Avg (salary)
1	-
2	-
3	-

## Problema sol

a)

(work)

b)

O Gnoce = 'Gomes', Fnoce = 'Corcos' (é plorée)

c)

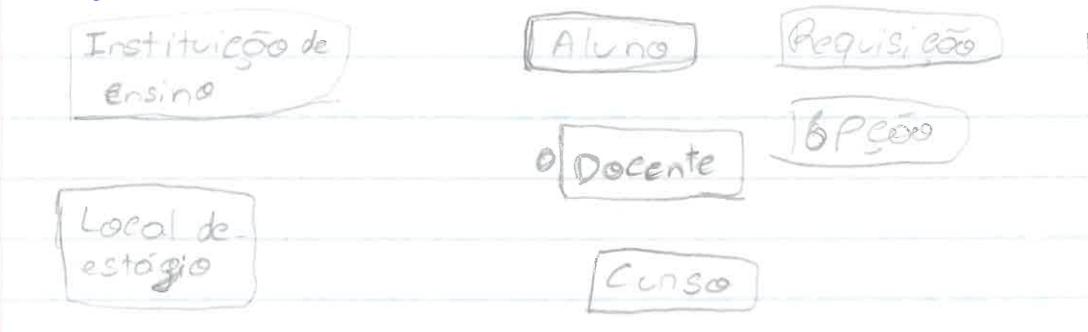
explorée 3 count (hours) (work-on)

d)

Idiotico.

## Projeto Locais de estágio

Entidades:



Aluno:	- Nome	- Ano
	- N° NEC	= 68888
	- N° ID Civil	
	- Endereço	
	- Contato	

Local de estágio:	- nome
	- Endereço
	- N. Registro Nec
	- N.º de vagas ->!
	- Contato

Curso:	- N.º ID
	- Nome

Docente:	- Nome
	- N.º ID Civil
	- Endereço ->?
	- Contato

Instituição de Ensino:	- Nome
	- Endereço
	- N.º Registro
	- Contato

## Linguagem SQL - DML

A partir da aula passada vamos mapear os conceitos para SQL

INSERT INTO "tablename" ("name", "ID", "Date nasc")  
se não temos os valores todos

Pode-se

especificar os atributos

INSERT INTO "tablename" (One, None, ...,  
(e, sorge, ...))

DELETE FROM "tablename" WHERE "Atributo" = X

também se pode especificar condições  
lógicas

Operações em conjuntos

SELECT \* FROM "TableName";

todos os tuplos onde os valores que seguem  
não são repetidos

ALL SELECT FROM EMPLOYEE

DISTINCT " " " "

↳ não há repetições

Renomeação é uma operação importante para  
eliminar a ambiguidade

SELECT [Fname + ' ' + Lname] AS Name

## Produto Cartesiano

SELECT \* FROM EMPLOYEE, DEPARTMENT;  
↑ colocaemos restrição

junção com Ambiguidade e → renomeação

SELECT ~~E~~ E.Fname, E.Lname, S.Fname, S.Lname  
FROM EMPLOYEE AS E, EMPLOYEE AS S  
WHERE E.Super\_SSN = S.SSN

Neste caso temos de eliminar a ambiguidade

## Tratamento de NULL

NULL → valor desconhecido ou não existe

As funções de agregação normalmente ignoram o NULL, para renificar usa-se o ISNULL

Em segue SQL os comparações lógicas com o valor NULL → resultam em UNKNOWN

## Nova regras do JOIN

```
*** FROM
WHERE EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber
WHERE Fname='Ricardo';
```

## Agregação

Dno → max(sal) = avg(sal) →

SELECT  
FROM  
WHERE  
Group By

COUNT(\*) → Nº de linhas

COUNT ("Atributo") → Conta o nº de vezes que aparece o atributo sem estar a NULL

Após a agregação, se desejarmos filtermos o resultado da agregação, não pode ser com o WHERE

Há a possibilidade concatenar queries em que o SQL pega num tuplo, → come o subquery e → retorna algum valor desejado.

## SQL - VIEW

Apresentação de dados os utilizadores, devem ser separados negros de prioridade.

→ Executam queries que podem ser mais ou menos complexas.

Há utilizadores que usam views de maneira exceciva, (tal como uma Maestro)

Perguntas sobre o soin e no agressão

SELECT Pname, sum(hours)

FROM Employees Projects → View: virtual

## Group By Name

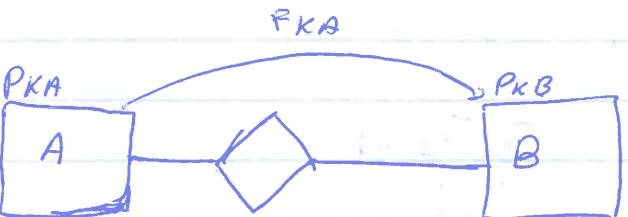
Tnota como subqueries são executadas substituindo o código, isto faz com que o desempenho seja afetado devido ao parsing.

Poderemos fazer inserts sobre as vias mas temos de ter atenção em quais queremos fazer insert, ~~é~~ é uma questão que não seleciona chaves ou NOT NULL não podemos fazer

Atenção sobre esta situação

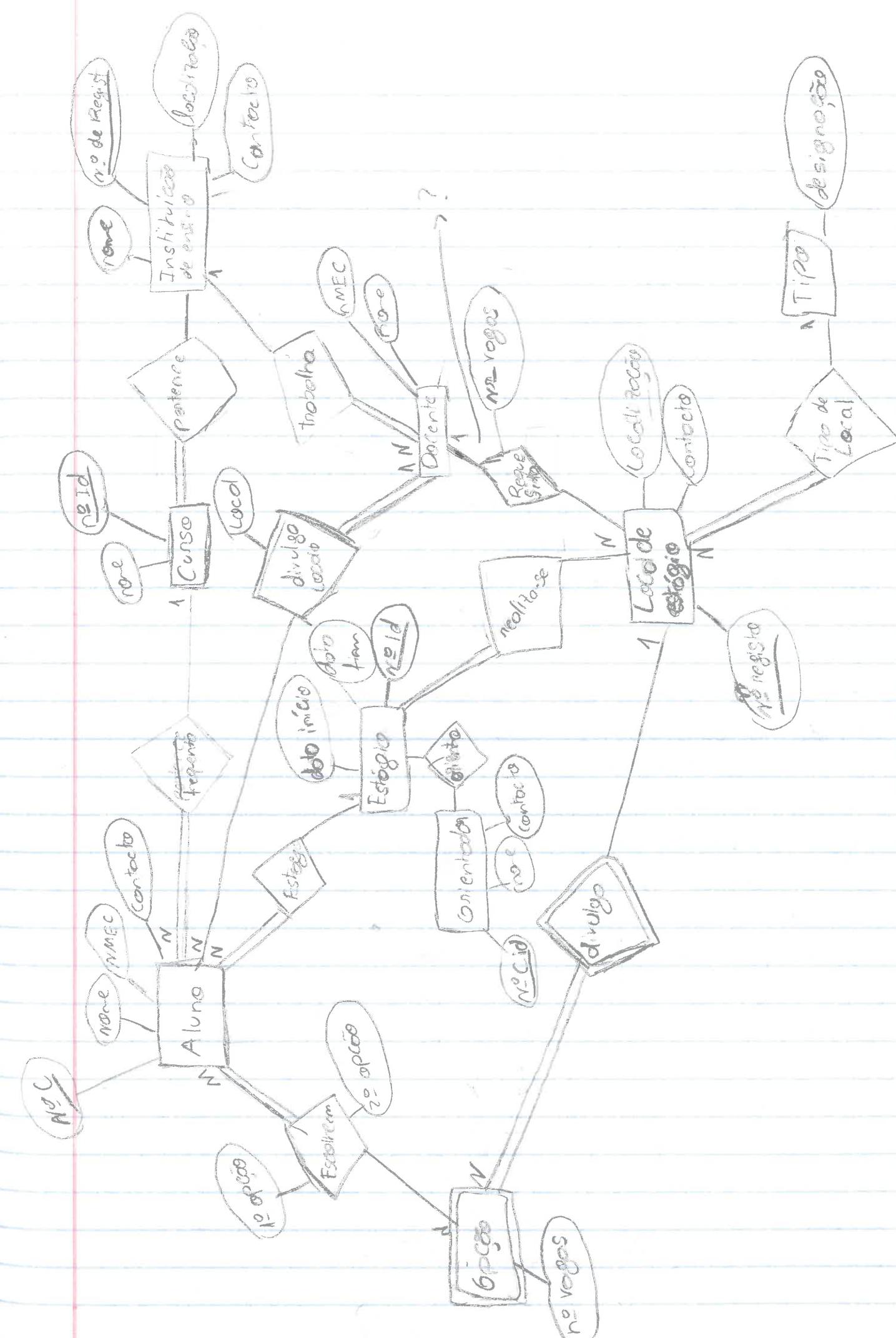
Deveremos usar a opção WITH CHECK OPTION para que ele verifique as condições das vistas ~~para~~ dessa maneira não avise se as alterações falam sentido.

## Normalização

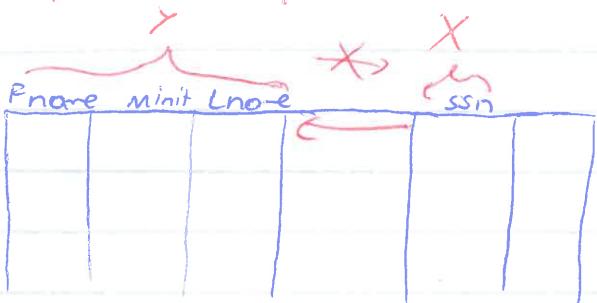


PKB	FKA
	✓
	null
	null
	null
	✓

como não obrigar todos a  
usa - se a regras ANM



## Dependências funcionais



para aquele Ssn temos associadas apenas aqueles nomes

Teste:

Construção de uma base de dados:

- 1º passo - Análise de Requisitos (txt)
  - Levantamento de informação
  - Filtragem de ruído

## 2º Desenho concepcional

- modelação e tratamento da entidades
- mapeia as entidades e relações do mundo real
- uma visão abstrata da estrutura da base de dados

## 3º Modelo Relacional Entidade/Relação

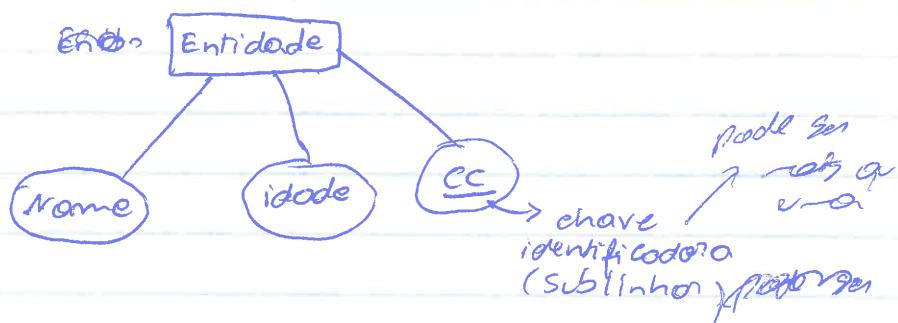
- Entidades - algo que existe

~~ex:~~

Entidade → Retângulo ~~Retângulo~~

Disciplina

Atributo



## Entidades

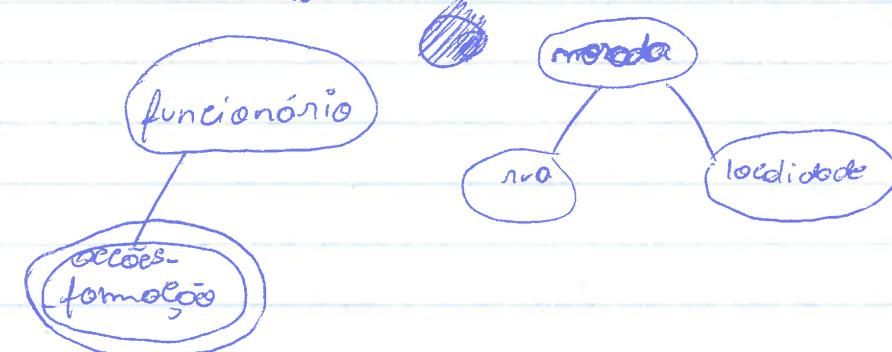
- Fontes = não dependem de outros para serem identificadas

- Focos - dependem de outros para serem identificadas



- Derivados → mudam ao longo do tempo

• Multivalor



Relações - entre duas ou mais entidades



também podem ter atributos

## Relacionamentos

- Grupo - no entidades na relação

- ~~unária~~ unária

- binária

- Ternária

## 6º Brigatoredade

Cardinalidade - nº de ocorrências

## • Relações neeunárias

Simétrica - não distinção

assimétrico - não distinção entre papéis

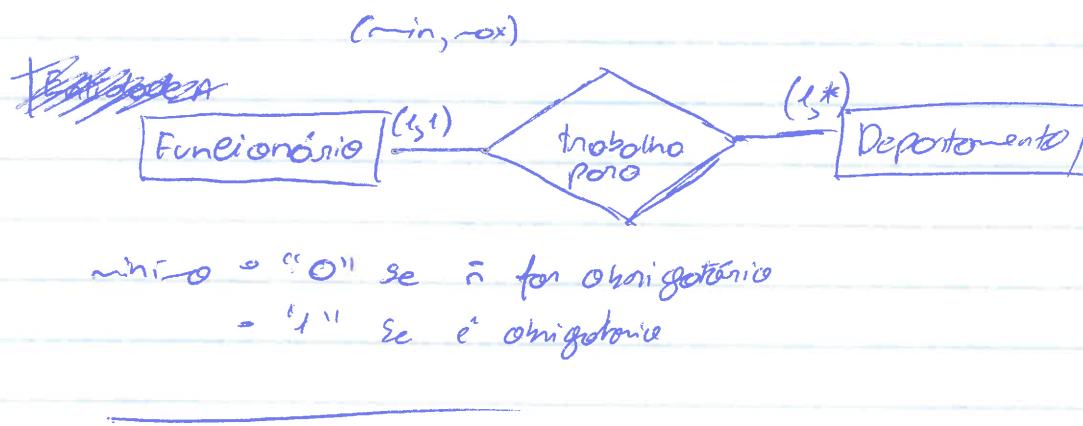


Objetividade

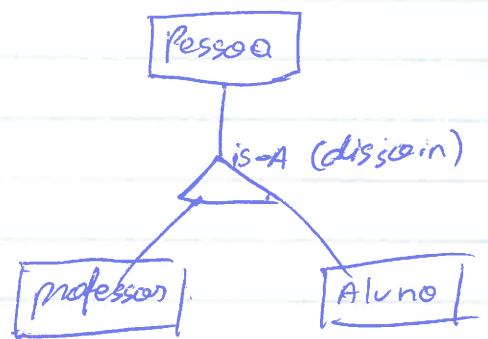
Linha dupla - a entidade participa em todos os na relação

" " singular - a entidade só pode ou não participar

notação alternativa:

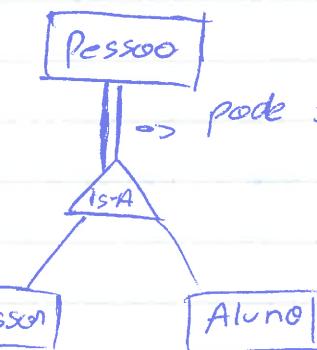


Um atributo pode generalizar ou ser especializado



Disjuntos: Entidade só pode pertencer a uma especialização

Sobreposta - pode ter pai de que uma especialização



→ pode ser de participação total

## • modelo Relacional

- modelo baseado na teoria dos conjuntos
- representado por tabelas

Esquema da Relação

• nome e atributos

Pessoa (nom, bi, idade)

Tuplo = linha de uma relação

	Atributos	
atributos	nmec	Nome
tipos	73941	Pedro
	02011	Maria
	7910	Tome

} cardinalidade

} grau da relação

Exemplo de Esquema da relação - Estudante (nmec, nome, curso)

Chaves

superchaves - conjunto de atributos que identificam de forma única os tuplos

chave candidata - subconjunto das superchaves que não pode ser reduzido sem perder a classificação de chave

chave primária - é a chave AD' atributo ou atributos selecionados

• de entre os ~~os~~ candidatos para o

"único" - chave candidata não selecionada para primária

chave Estrangeira - chave importada de outra relação

No escolha da chave primária temos de ter atenção ao fato de essa chave pode ser alterada à posteriori, aqueles que não foram escolhidos tornam-se chaves ~~o~~ unicas

### Integridade

Dore ser garantida pelo SGBD

• Dominio - nomeiros mols elementos

• Os campos devem obedecer às restrições impostas. chave primária

Ex: NOT NULL

~~chave~~ Sexo Chor CHECK(Sexo='m' OR Sexo='f')

• Entidade - todo tuplo deve ser identificado de forma única

• Referencial - A chave estrangeira ou é null ou então uma chave primária de outra tabela

### Regras de Codd

verificam se ~~o~~ efetivamente uma base de dados pode ser chamada de relacionais ou não. É um conjunto de 12 regras, o próprio codd admite que é difícil cumprir todas.

### Conversão do modelo ~~relacional~~ DER em modelo relacional

#### Passo 1:

para cada entidade forte elenca uma tabela com os seu atributos.

#### Passo 2:

Para cada entidade fraca importa como chave estrangeira a chave primária de entidade dominante, assim a chave primária não ser a combinação da chave estrangeira e g do parente da fraca.

#### Passo 3:

Em relações 1:1 escolhemos uma tabela em que o sua participação seja total e importamos a chave primária da entidade com que se está relacionada, quaisquer outros atributos da relação são também importados.

Em caso de não existir uma ~~um~~ um dos lados com participação total trata-se como uma relação de 1:m.

#### Passo 4:

Para relações 1:N importa-se como chave estrangeira, ~~o~~ na tabela do lado com ~~N~~, a chave primária da ~~do~~ entidade do lado com 1. Quaisquer outros atributos do relacionamento também são importados para essa tabela.

### passo 5:

Para as relações N:M é criada uma nova tabela com os chaves primárias das entidades adjacentes.

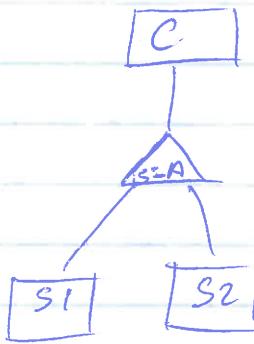
### passo 6:

Atributos devem fazer ponte de uma tabela com o designação do atributo e como chave estrangeira a chave primária da entidade na qual 3 atributos. A sua chave primária inclui a combinação da FK e da designação do atributo.

### passo 7:

Relacionamento n-ário ( $N > 2$ ), deve-se criar uma tabela nova imprimindo os chaves PK das tabelas adjacentes.

Em caso de especialização:



Pode-se fazer de duas maneiras:

- 1º Cria-se uma tabela para a super classe e outras para os derivados que partem com chave estrangeira a chave da super classe. Funciona para casos de especialização total e parcial.

2º Cria tabelas para cada uma das classes derivadas. Incluindo

os seus atributos e os da super classe.

Apenas funciona para especialização total.

### Employee

SSN	Belie	Frame	Miait	Lname	Address	salary	Set	Sup-SSN	Prnum
-----	-------	-------	-------	-------	---------	--------	-----	---------	-------

1º

### Department

Number	Name	Mgr-SSN	Mgr-start-date
--------	------	---------	----------------

### Dept-Location

Location	Prnumber
----------	----------

### Project

Number	Name	Location	Prnumber
--------	------	----------	----------

### Dependent

Name	Essn	sex	Birth date	Relationship
------	------	-----	------------	--------------

### works\_on

Essn	Prnumber	Hhours
------	----------	--------

## Linguagem SQL

Diz-se em ~~as~~ duas partes

DDL - Data definition Language

DML - Data manipulation

## DDL

CREATE DATABASE "databaseName"

DROP DATABASE "

agrupamento de tabelas

CREATE SCHEMA "schemaName"

②

domínio

CREATE DOMAIN comSalario INTEGER  
NOT NULL CHECK (comSalario > 475)

/

Não existe em SQL

CREATE TYPE SSN FROM Integer NOT NULL  
CHECK (SSN > 0);

ALTER TABLE "tablename" DROP COLUMN  
"CONSTRAINT"  
ADD "

ALTER TABLE "tablename" ALTER COLUMN "columnName" CHAR(9)

constantes de 9

conectores

## Algebra relacional

União - ocorre entre duas tabelas com o mesmo nº de atributos e com o ~~domínio~~ o mesmo domínio.

U ↳ Resulta todos os tuplos exeto duplicados

Interseção - necessita das mesmas condições que o união e resultam os tuplos iguais em ambas tabelas

∩

Diferença - mesmas condições da união mas devolve os tuplos que aparecem se encontrarem fora da interseção das duas tabelas.

nomes dos funcionários que são gestores de departamentos

MAN — DEPARTMENT ~~man.ssn = ssn~~ EMPLOYEE

Junção

IT phone, name, lname (MAN)

Equijoin

Nextendjoin

Faz a junção dos colunas com o mesmo nome

Nota: Por vezes realiza-se o renomeação para facilitar a junção.

~~Div~~

Divisão

6. o resultado incluirá todos os tuplos que estiverem relacionados com os atributos usados na comparação

R	S	T
A a b c d e f g h	C d b	a b

:

~~Inner join~~  
os tuplos sem correspondência são descartados  
~~OUTER~~

semi join

Left semi join

R	S	T
x a b	y d e	x y b
z c d	z g h	

Faz a representação dos tuplos em junção natural dos tabelas R e S mas mostra os tuplos de R

Right join

6. o comportamento é semelhante ao de cima mas faz a representação do coluna do tuplo do tabela do direito, S.

Inner join

os casos anteriores são inner não representam os tuplos sem associações

~~Outer join~~ Outer join

Faz a representação de todos os tuplos, mesmo aqueles sem associação.

Left Outer join

é representado os tuplos da tabelas da esquerda juntamente com os associados, os sem associação são representados na tabela

Agregações

sum: soma dos valores

count: contagem dos valores

ο farmaco = null (FARMACO  $\Delta$  código = fármaco Prescreve)

num\_consulta count(Fármaco) (Consulta  $\Delta$  número\_consulta: consulta Prescreve)

top - Π T1.codigo\_consulta, num\_farm = count(fármaco) (Prescreve)

Π T1.codigo , media\_farm\_consulta = avg(num\_farm) (Top)

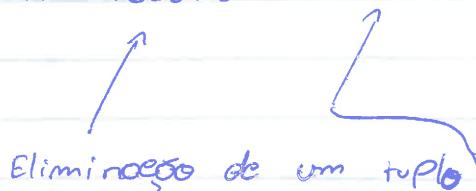
## SQL DML

Insert INTO "tablename" VALUES ( ... )

/

inserei valores numa tabela

DELETE FROM "tablename" WHERE "Alguma condição"



Sem WHERE, eliminam-se todos os tuplos

UPDATE PROJECT

SET Plocation = 'Bellville', Dno = 5

WHERE Pnumber = 10;

/  
atualiza um tuplo

UPDATE EMPLOYEE

SET Salary = Salary \* 1.1;

WHERE Dno = 5;

/  
afeta n - tuplos

ALL → seleciona todos os tuplos, mesmo os duplicados

DISTINCT → Apesar de ~~não~~ Sem repetições

$U = R_1 \cup R_2$

$\cap = R_1 \cap R_2$

$- = R_1 \setminus R_2$

diferença

Produto cartesiano

SELECT \* FROM "tablename1", "tablename2";

Comparação de strings

SELECT \*

FROM PROJECT

WHERE PLOCATION LIKE '%Houston, Texas%';

zero ou mais caracteres

caractere de comparação

... LIKE '\_ - S - - G - - %';

/ um qualquer caractere

WHERE (Salary BETWEEN 3000 AND 4000)

/

Teste do ano passado

- Q1 a) V                            Q2 a)
- b) F
- c) V
- d) V
- e) F
- f) V
- g) F
- h) V

**CLIENTE**

num-corta	endereco	nome	NIF
-----------	----------	------	-----



**ALUGUER**

duração	número	data	CLI-NIF	num-Balco-mon-va
---------	--------	------	---------	------------------

**BALCAO**

nome	endereco	número
------	----------	--------



**VEICULO**

matricula	ano	marca	Cod-tip
-----------	-----	-------	---------



**Tipo-VEICULO**

designação	oncondicionado	Código
------------	----------------	--------



**LIGEIRO**

numlugares	pontas	composiç rel	Cod-vei
------------	--------	--------------	---------

**PESADO**

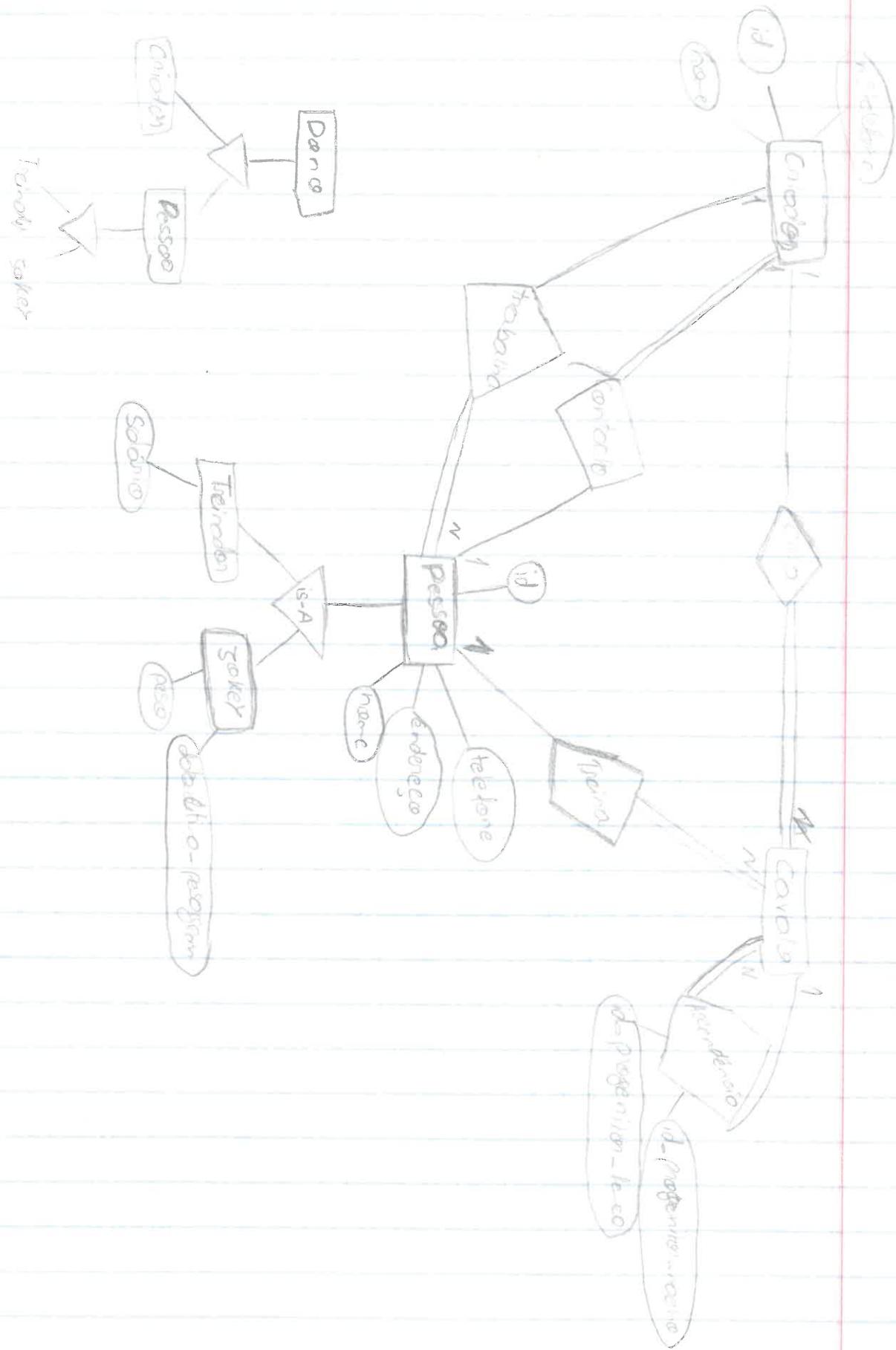
peso	passageiros	Cod-vei
------	-------------	---------

**SIMILARIDADE**

cod-tipo1	cod-tipo2
-----------	-----------

critico  $\leftarrow$  km  $\rightarrow$  especie, todo

km	id espe	id critico
----	---------	------------



## View

- Relação virtual
- São representações dos dados de tabelas, não segundo armazenamento de dados
- É possível manipular dados das views
- São utilizados em associações de esquemas de bases de dados e diferentes aplicações
- Também são usados para tratar de questões de segurança e integridade

CREATE @ VIEW "some-name" AS

SELECT name, Fname

FROM PEOPLE

WHERE (Salario BETWEEN 3000 AND 4000);

Após criação de uma view é possível efectuar operações sobre ela.



### nested view

- views que usam outras views como fonte

É possível inserir dados na view com o insert into mas por vezes não fazem sentido, quebrando assim a integridade de dados. Para evitar essas situações devemos colocar o comando WITH CHECK OPTIONs

ou ALTER VIEW "name" AS

CREATE VIEW "name" AS

SELECT Fname, Dno

FROM EMPLOYEE

WHERE Dno = 5

WITH CHECK OPTION;

Note neste caso, se

tentarem fazer

INSERT INTO "name"

VALUES ('sose', 8)

/  
emo

## Verificação de lógica

Projetos queríveis:

Listas de opções para o Aluno

Listas locais de estoques para o Professor  $\rightarrow$  Person -> Person -> Person

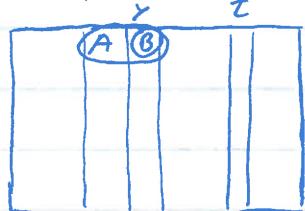
Estoques por Aluno

dnumber  $\rightarrow$  dName

Sempre que um certos dnumber aparece, tem que estar associado a só nome

Axiomas de Armstrong

- Reflexibilidade :  $Y \subseteq X \Rightarrow X \rightarrow Y$



- Aumento  $X \rightarrow Y$  em  $XZ \rightarrow YZ$

X	Y	Z
1111	John	100

$$X \rightarrow Y$$

$$XZ \rightarrow YZ$$

- Transitividade

$$X \rightarrow Y \text{ e } Y \rightarrow Z \Rightarrow X \rightarrow Z$$

- Dependência total

Atributo depende do chare

- Dependência parcial

Atributo depende de um parte da chare

muito mais importante

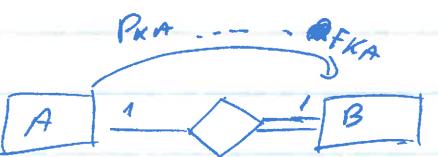
Préima Forma Normal (1NF)  $\nearrow$  um projeto

Um atributo apenas deve conter um campo, ou seja, um nome deve ser dividido num atributo Fname e Lname ou morada em sua , no ponto, Freguesia.

## Análise de Qualidade

Critérios Informais: - Redução de null

...



Pq n procede o contrário?

para reduzir os NULLS, se A n se relaciona com B o em  
campo iria estar a NULL

se caso num caso de una rel. se obrigatoriedade enio-se una tabela nova  
que n o haverá possuir NULL, se existe a relação, existe tuplo

Dependências Funcionais

Relacionamento

$$X \rightarrow Y$$

X define Y ou Y depende de X

Ex:

$$\text{ssn} \rightarrow \{\text{Fname}, \text{Lname}, \text{Salary}, \dots\}$$

ssn define de forma única  $\nearrow$

## Segunda Forma normal

Eliminar dependências parciais  
vai exemplo da aula

• numa tabela queremos eliminar a redundância, se na tabela tivermos o ssn de funcion. e number de projetos. Funcionário que trabalha no mesmo proj. iria fazer aparecer o number n vezes.

## Terceira Forma normal

Elimina dependência transitiva



Há mais formas normais normais, até à terceira é só vantagens a partir dessa para a frente é necessário ter cuidado pois só começam a crescer e se desvantagens.

BCNF → Boyce-Codd-Normal Form

↳ Ainda não percebido → ver exemplo

A 4NF e a 5NF 5 nodos

## 4NF Dependências Multivárias

X	Y	Z
$x_1$	$y_1$	$z_1$
$x_1$	$y_2$	$z_2$
$x_2$	$y_1$	$z_1$
$x_2$	$y_2$	$z_2$

→ Podemos decompor  
Pode-se usar algoritmos para detectar

## 5NF → Dependências de junção → Ainda mais nora

Pode-se decompor as dege tabelas e por dependência de junção é possível economizar algum espaço.  
Fazendo a junção das tabelas de maneira a obtermos a original estarmos na quinta forma normal.

## Indexação e optimização

• É mais eficiente fazer uma seleção/projeção de dados secessorios e de seguida fazer as junções do que juntar tudo num tableau e de seguida fazer essa seleção/projeção.

```
SELECT Fname, Lname, Patient_ID
FROM Patients
WHERE Fname='Maria' AND Lname='Pinto'
```

• SGBD abre todos os tuplos e procura 1 a 1? → O(n) de complexidade

Isto não é executável se o SGBD não estrutura de dados

Índices → estruturas de dados que permitem pesquisas, têm um ponto de referência para a localização física

• N Índice: Início da página

• Índice: offset para a localização do tuplo

Vantagem → Mais eficiente na pesquisa

Desvantagem → Também ocupam espaço e na inserção de dados tem que haver uma inserção na tabela e uma atualização.

Nos parâmetros dos índices podemos estipular o nível de complexidade de procura ex: procura por nome Matto e procura por nome Pinto



Single-level → Armazenam ~~no~~ o valor do atributo indexado e a sua localização

Primary index\* → Indexa atributo chave da relação

Cluster index\* → Indexa atributo que não é chave, resulta normalmente em raias tuplos, ou seja, é denso, não havendo tuplos a serem referenciados por esse índice.

Secondary index\* → Valores duplicados que seja necessário indexar de maneira ~~no~~ mais específica cria-se um índice para um índice que aponta para o tipo

8.1

a) Os atributos são indivisíveis

b)

Cade

B | A | D |

2FN  $(R1(A, \underline{B}, D, E, G, I))$

$R2(B, C, D, E, F, G, H)$

CCQ

3FN  $(R(A, \underline{B}, D, F, G, I))$

$R(\underline{D}, E, F)$

$R(\underline{G}, H)$

$R(\underline{B}, C)$

Scannable

8.2

a)

$R = \{A, \underline{B}, C, D, E, F, G, H, I, 5\}$



A chave é o atibuto

A e C

b)

$2FN$   $\left\{ \begin{array}{l} R_1(\underline{A}, \underline{B}, C) \\ R_2(A, D, E) \\ R_3(\underline{B}, F, G, H, D, I, J) \end{array} \right.$

b)

$2FN = R(\underline{A}, \underline{B}, C, D, E)$

$3FN$   $\left\{ \begin{array}{l} R(\underline{A}, \underline{B}, C, D) \\ R_2(D, E) \end{array} \right.$

c)

$3FN$   $\left\{ \begin{array}{l} R_1(\underline{A}, \underline{B}, C) \\ R_2(A, D, E) \\ R_3(\underline{B}, F) \\ R_4(E, G, H) \\ R_5(\underline{D}, I, J) \end{array} \right.$

c)

$BCFN$   $\left\{ \begin{array}{l} R_1(\underline{A}, \underline{B}, C, D) \\ R_2(D, E) \\ R_3(A, \underline{C}) \end{array} \right.$

8.3

a)  $R = \{A, B, C, D, E\}$

A chaves são os atributos A e B.

8.4

a)  $R = \{A, B, C, D, E\}$

A chaves são os atributos A e B.

- b)  
c) 2 FN { R<sub>1</sub>(A, B, D, E)  
d) 3 FN { R<sub>2</sub>(A, C)  
BCFN

8.5

a)

Pessoal(nid, contacto, nome)

Aluno (n\_id, ano-curricular, nomec, idun, ~~noz~~<sup>desjso</sup>)

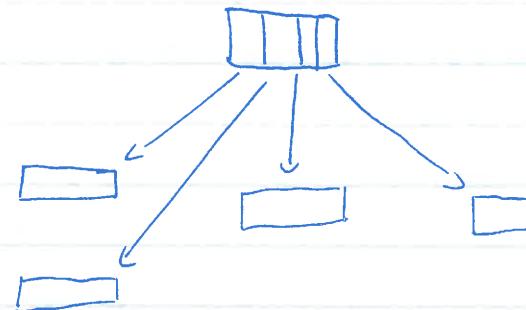
Socid{ —> Sono, ... }

Índices - São muito importantes no nome de base de dados, e os que têm melhor frente de rendimento

Aceleram processo de pesquisa mas também ocupam espaço em disco

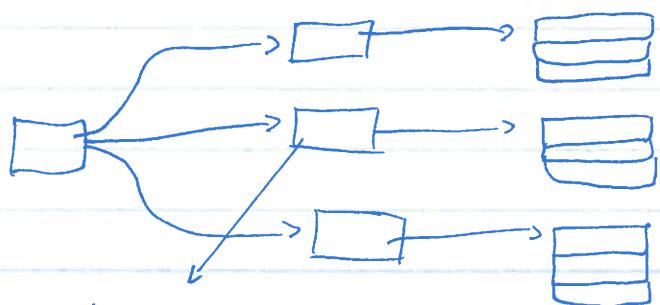
Secundário index :

### Multilevel Index



podemos ter vários níveis de indexações, tornando a busca mais eficiente para a complexidade da busca é reduzida

Um problema associado com esta estrutura é uma nova indexação que tenha de ser feita no índice que só está cheio vai obrigar a fazer page split. Isto é, os índices têm de ser copiados para outros índices para alocar espaço



desejamos adicionar

uma indexação se só estiver cheio teremos de alocar espaço

### SQL - Index

```
CREATE INDEX idx_PName ON Project(Pname);
podemos ter índices com mais do que um atributo
" " " idx_Employee ON Employee(Fname, Minit, Lname);
```

- Ao desenvolver interfaces temos de ter consciência do que da implicações que isso traz para a base de dados

↳ "Este conhecimento vale dinheiro"

Numa experiência:

	total de registros	I/O	Records Retrieved	Cost	Reads
①	72 591	• 485	526	• 485	→ Leitura de todos
②	9	• 003	2	• 003	→ Leitura com índices
③	11 05	• 004	4	• 004	

2 leituras podem originar de dois níveis de índices



Deve-se evitar operações de I/O, um bom índice evita isso

### Moderação da indexação:

Temos de evitar índices que têm valores repetidos

Ex: pesquisa de uma pessoa com índice que aponta para o Conselho onde ela vive. Ima resultar em muitos tuplos

A escolha do índice deve atender a alguns compromissos! Consulta no slide 18 Índices

Há situações em que não sabemos em que contexto foram desenvolvidas certas queries, índices, etc. Se pretendemos ter uma ideia de como é usada podemos usar benchmarks para verificar o histórico.

• base de dados funcione  
Temos de garantir que a partir do momento zero temos de garantir que a base de dados está pronta para suportar dados em grande quantidade, ou seja eventualmente temos de preparar as nossas opções para qualquer eventualidade que possa ocorrer durante a sua utilização a futuri posterior. Vai de pensar só no momento de criação.

### SQL Server - Indexing

Tipos de índice:

- Clustered → A estrutura de dados está associada ao nó de pesquisa

Neste caso só é possível ter um índice clustered, não tem ponteira para outra estrutura, ver slides.

~~mas podemos ter outros índices secundários~~  
~~para outras estruturas~~

• Non-clustered → A indexação não é feita pela chave primária mas por outros atributos até chegar à chave primária e depois é possível ir à chave primária estrutura principal para buscar a informação.

## Page split

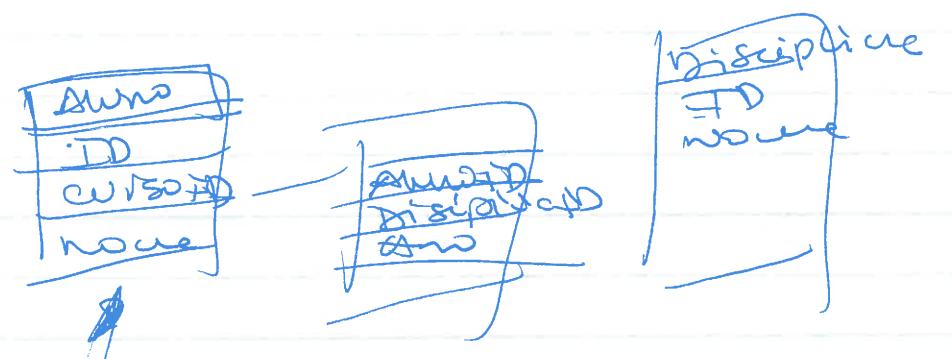
Em clustered index n tem problemas de page split, se a chave PK for um nº ou algo sequencial.  
Ao contrário dos não clustered.

~~Page lock~~

Para evitar isso temos **FILL FACTOR**, **INDEX PAD**

Podemos definir até que percentagem queremos encher os nossos tabelas reduzindo os page split mas despendendo espaço

se queremos fazer Fill Factor nos tabelas intercaladas

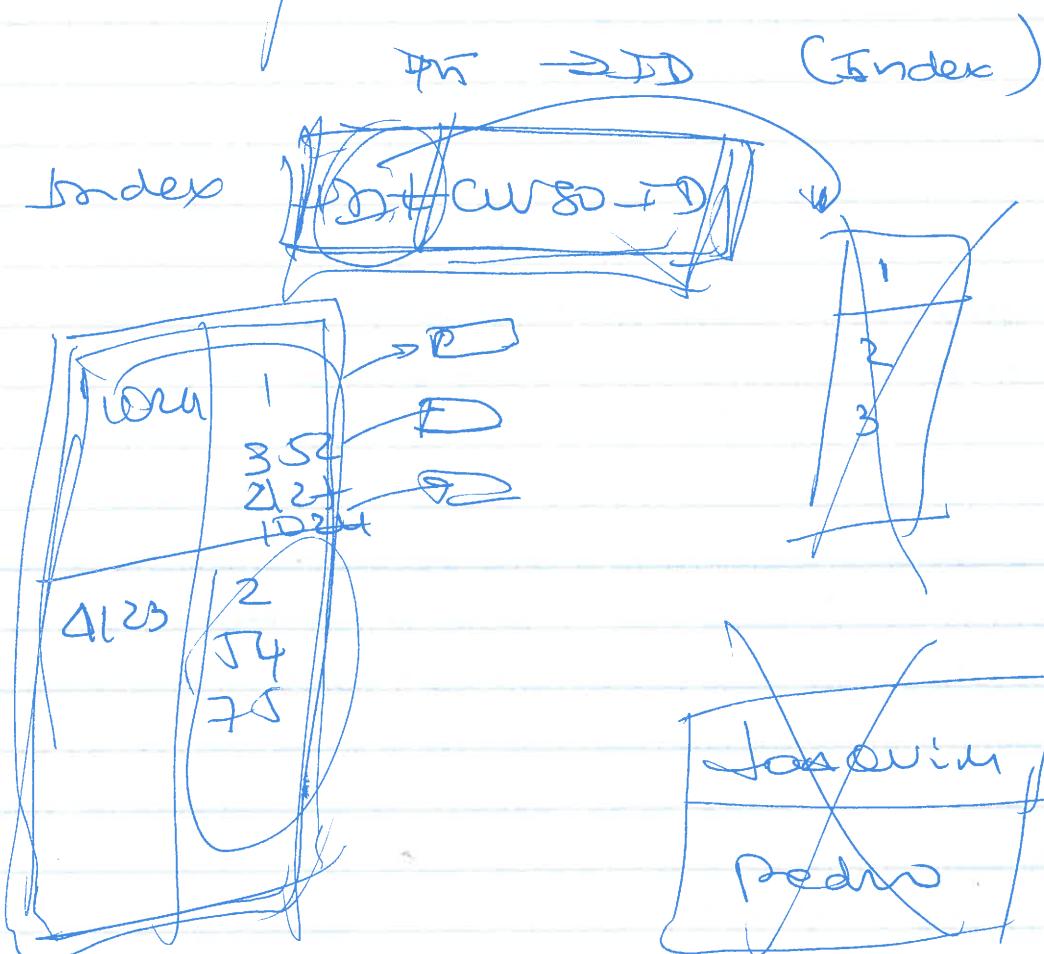


## Programação em SQL

Pensando na tabela Emploee com vários dependências

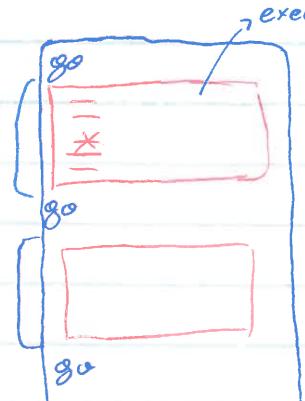
Temos de pensar se no contexto das nossas actuações a pena eliminar o funcionário ou meter ativo ou n

Para eliminar o Emploee temos de verificar todas as suas dependências e só se n nenhuma tiver é que pode ser eliminado

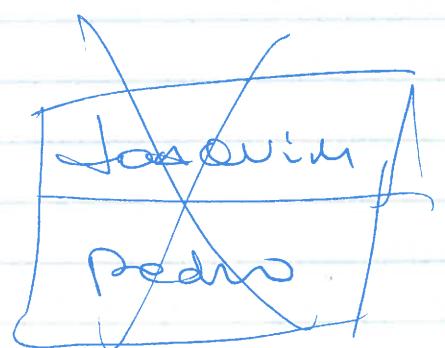


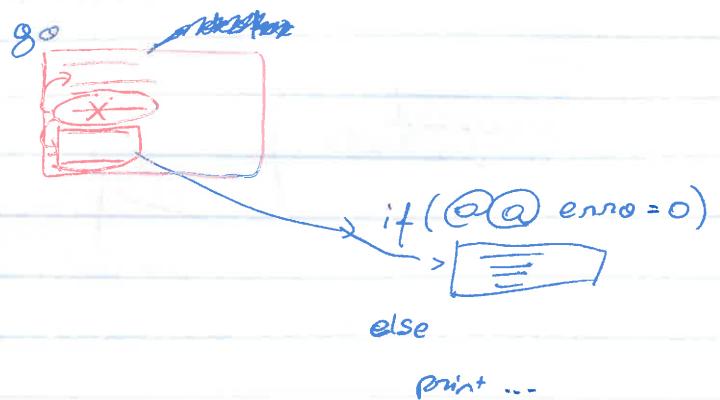
**Batch** → Grupo de instruções SQL que constituem uma unidade lógica

go → delimita uma batch



evita que, num n elevado de de intru., se uma tiver um erro, ate n pára devido a esse erro





⑩

~~Procedimento~~

Variáveis

Declarar @x valorchar('0') = '666'  
" @dy as int  
↳ opcional

Atribuição de valores

SET @x = 'António';  
SET @y = s;

poderemos atribuir um valor  
logo no momento de  
programação

~~poderemos~~ ou

SELECT @price = price FROM Product  
WHERE ID = .vn

NVARCHAR(...) → codificada em

unicode

Display de variáveis

SELECT @x  
OU  
Print 'varx: ' + str(@x);

cada Batch tem um scope, variáveis declaradas num batch  
não são reconhecidas noutras

Nota:

Evitar key words como ORDER na  
nome da tabela

CASE ... WHEN  
↗ switch

Tabelas Temporárias

permitemos armazenar mais do que um valor

CREATE ?

⑩ DECLARE @Temp TABLE (PK INT PRIMARY KEY, ...)

INSERT INTO @Temp --

⑩ ♦ não podem ter associações de chaves estrangeiras  
Também está limitado pelo scope do batch se forem declaradas  
como variáveis

DECLARE #Hello

⑩ # afirma que a tabela não está limitada  
CREATE TABLE #Hello pelo scope do batch

Mas está limitado por sessões

Também podemos criar uma global, comum a outras sessões

CREATE TABLE ##Hello

estão localizadas num  
diretório # do rota de BD

## Cursors

### Cursors

São muito exigentes em termos computacionais.

Só instruções executadas sequencialmente

Ex: usando uma query de loop-se para cursor e executar-se o cursor, por cada vez que vai ser executado vai iterar sobre o cursor retornando os resultados após cada iteração

Têm que ser abertos, fechados, e a sua memória tem de ser libertada

Só devemos utilizar os cursor quando necessário  
ver situações recomendáveis

É útil ao trabalhar em equipas

um procedimento é mais eficiente pois fica armazenado em cache e apenas tem que ser executado, não requer compilação

Ex: de utilidade → num disjuntor podemos garantir a integridade de dados, criando um procedimento para adicionar um tuplo fazendo a verificação se ele existe na outra tabela.

### Precisar os mais rápidos dos procedimentos

Podemos tratar isto → como classes em programação aplicando as regras de programação funcional. O cliente só precisa de mexer nos procedimentos e não deve ter manipulação direta

Os procedimentos não estão associados a uma tabela

## Funções

Qual a diferença entre funções e procedimento?

Têm as mesmas vantagens → em termos de performance do que os procedimentos.

Nunca podemos usar → SP como fontes de dados

exec mySP...

select \* from → SP → com as funções é possível fazer isso.

## Procedimentos

Para eliminar um funcionário → teríamos que apagar todas as instâncias de determinado funcionário na tabela WORKON. Portanto → , devido às suas SSN e SÓ depois de eliminar essas dependências é que podemos eliminar um tuplo do funcionários

→ Todos estes procedimentos podem ser implementados num procedimento, assim numa simples operação DELETE faz todos estes procedimentos

Existem 3 tipos de funções:

Escalares: Retornam escalares

Ex: variáveis do tipo money, int, varchar

Úteis em CHECKS com uma lógica complexa

Em vez de usarmos views podemos criar views parametrizáveis  
Assim temos uma maior capacidade operacional

### Triggers

↳ São procedimentos que são ativados quando ocorre determinada operação

Ex: para garantir um disjoint entre tabelas, podemos usar um trigger que faça o tratamento dessa situação.

Podem estar associados a DDL e DML

! só ramos tratam

Numa operação de inserção, é feito um conjunto de operações de verificação antes de fazer as operações

Triggers — ~~EXISTEM~~ INSTANCE OF

AFTER → é possível fazer rollback

Os triggers têm acesso às tabelas temporárias inserted, deleted nestas tabelas para fazer operações sobre elas.

As interações vêm

num trigger se chegam os à conclusão que afinal é para continuar com mais outra vez a operação mas na segunda opção o trigger não é acionado.

Só se verifica a validade mensal para para projeto

tabela explorar + tem comentários

↳ ver os funcionários no projeto e os respectivos salários

### Transações

Conjunto de instruções atômico, ou seja, todos executados, ou nenhum é

↳ há operações que não podem ser interrompidas a meio se tal acontecer deve ser feito o rollback.

poderemos definir as características da transação e definir quais têm mais prioridade entre elas sobre as outras.

Permite assim controlar o acesso simultâneo por vários utilizadores.

Se for feito rollback os operações desfeitas são apenas as que se encontram dentro da transação

### Características das Transações

- Isolamento → com vários clientes operando sobre os dados devendo garantir a exclusividade entre operações.

- Ver mais características

## Controle de Concorrência

Escolonamento → temos de evitá-lo para inconsistência de dados (leituras perdidas)

Leituras Sujas → folhos nas transações pode causar estas situações  
pois uma folha inicia original rollback limpando todos os dados anteriores mas se esse rollback for feito depois de outras transações terem operado sobre esse dados irá levar a inconsistência de dados.

Métodos:

- mecanismos de locking
  - " de etiquetas }
  - métodos optimistas → partem do princípio que as interações são raras
- preventivos

Etiquetagem → cada trânsito é etiquetado, quando há conflitos → voltam para a fila de espera

Locking → bloqueia o acesso ao registo quando precisa, em situações de deadlock o sistema libera os recursos e os trânsitos

## Recoverabilidade de folhos

Todos os sistemas computacionais estão sujeitos a folhos

Ex: - durante a escrita o sistema vai a baixa

- disco estraga-se por alguma razão

Porta é num hospital se a base de dados vai a baixa podem morrer pessoas e trazer-nos implicações legais

Gravidade: folha no transação → menos graves

Péda da base de dados → grave

As menos graves são tratadas pelo próprio sistema

Abort em cascata → perigo de perda grave de informação

Backups → cópias de segurança em caso de perda de informação

Desvantagem: só possuir info até ao ponto onde foi feito o backup.

Dever-se fazer backups com regularidade mas são perigosos e demorados

Podem ser feitos em horas mortas da atividade, é preferível

Transaction Logs → regista todos os operações realizadas na base de dados, este registo não deve estar no mesmo disco que a base de dados

Ex: Em caso de falha do disco podemos ir buscar o backup e de seguida podemos fazer o rollforward até obter os dados antes da falha.

usando as before image e after image verificamos se os outros alterações feitas antes e depois

Check point → demarca o momento até onde é feito o rollback ou rollforward

## Controle de Concorrência

Escolonamento → temos de evitá-lo para inconsistência de dados (Leituras perdidas)

Leituras Sujas → faltas nas transações pode causar estas situações pois uma falha irá originar rollback limpando todos os dados anteriores mas se esse rollback for feito depois de outras transações terem operado sobre esse dados irá levar a inconsistência de dados.

Métodos:

- mecanismos de locking
  - " de etiquetas }
  - métodos optimistas → partem do princípio que os interferências são raras
- preventivos

Etiquetagem → cada trânsito é etiquetado, quando há conflitos → voltam para a fila de espera

Locking → bloqueia o acesso ao registo quando precisa, em situações de deadlock o sistema liberta os recursos e os trânsitos

## Recoverabilidade de folhas

Todos os sistemas computacionais estão sujeitos a folhas Ex: durante a escrita o sistema vai a baixo

o disco estraga-se por alguma razão

Falta a um hospital se a base de dados vai a baixo podem mesmo faltar pessoas e trazer-nos implicações legais

Gravidade: falha no transação → erros graves  
perda da base de dados → grave

As erros graves são tratados pelo próprio sistema

Abort em cascata → perigo de perda grande de informação

Backups → cópias de segurança em caso de perda de informação

Desvantagem: só possível inf. até ao ponto onde foi feito o backup.

Dever-se fazer backups com regularidade mas não pesados e descoordenados

Podem ser feitos em horas mortas da atividade, é preferível

Transaction Logs → regista todas as operações realizadas na base de dados, este registo não deve estar no mesmo disco que a base de dados

Ex: Em caso de falha do disco podemos ir buscar o backup e de seguida podemos fazer o roll forward até obter os dados antes da falha.

usando as before image e after image verificamos se as alterações feitas antes e depois

Check point → demore com o reento até onde é feito o rollback ou rollforward

~~Sobre punto E permitenos~~

## Segusionea

Tod como num sistema operativo no só também roários  
objetos com permissões diferentes

No SQL há autenticação, podemos criar contas no SQL e no windows, só para logins no SQL Server

Os login podem estar associados a regras, ou administradores podem ter permissões adicionais, ao contrário dos usuários normais.

~~Esses utilizadores têm acesso a bases de dados distintas~~

No momento podemos observar ~~que~~ este vários logins e regras dos servidores, estas regras definem quais privilégios para tipo de utilizador tem. Um utilizador pode pertencer a vários grupos.

Escolhendo uma base de dados também temos os usuários e as regras

Podemos criar login mas devemos usar um sp- que serve especificamente para isso

Cada base de dados tem informações e também tem relacionados a relacionados com ela.

Temos uma grande quantidade views na master, mas só em qualquer base dados,

Poderemos consultar informações como tabelas criadas, foreign keys, indices, sp's, views, etc.



Na tabela Loco de Estágio inserir no Coluna nome ✓

↳ Table Aluno-Redator - estágio ↳ Redator(a) do  
cian view estágio

No topo estágio alt. com espécies orientadas para o topo. Proliferação estágio v

criar trigger para o insert de sp escolhos  
ex: só pode inserir se é dia de aula  
verifica se só tem um tupla ✓

criar sps que usam hash no lado do aplicação ✓  
• através das views podemos ver colunas, filtro-las  
por tabelas, etc

Para um aplicação que faz desse a base dados haverá  
um user que terá certas permissões

• É mais fácil criar perfis com permissões pre-definidas  
e de seguida adicionais a utilizadores a contas perfis.

Temos de ter cuidado pois podemos ter permissões para executar  
um SP mas dentro desse SP não há operações que não tenha  
permissões para executar

### SQL Injection

Não podemos fazer a manipulação de código direto do lado  
da aplicação misturando com objetos do código do outro  
lado, torna a base de dados vulnerável a ataques

Ver possíveis nos slides

É possível evitá-lo algumas maneira restringindo excessivas permissões  
de acesso mas adobaria por trás também funcionando dados, para  
isso devemos usar outras técnicas.

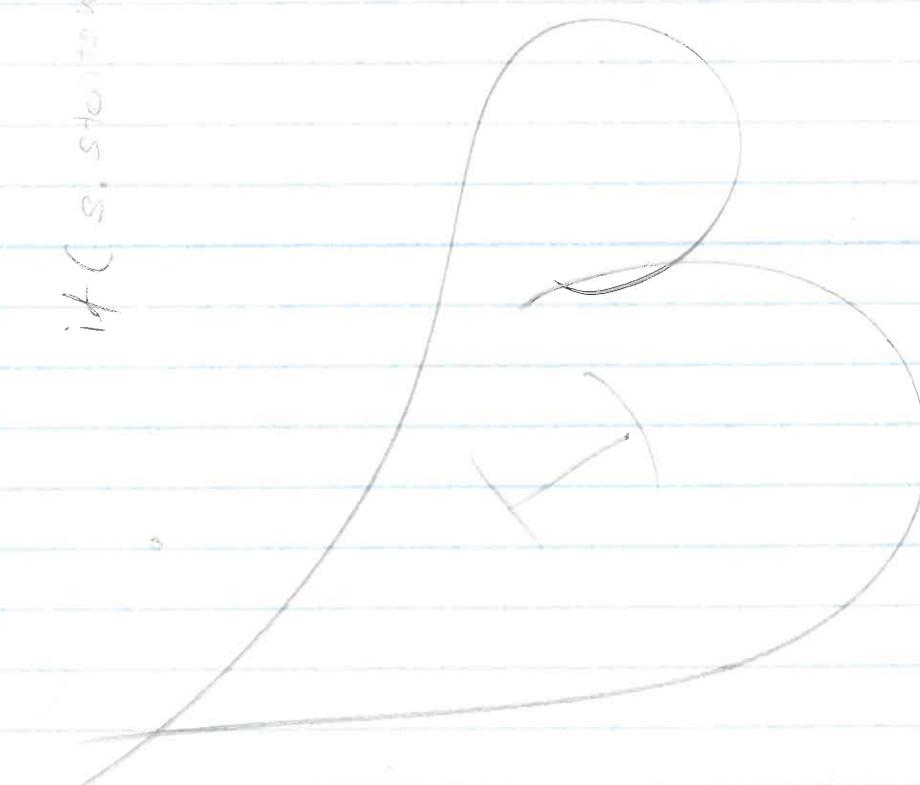
Descontos sempre nos dados inseridos pelo utilizador → limpeza de caracteres  
Nunca utilizar SQL Dinâmico verificando os caracteres  
perigosos

Código pode haver injeção ↗

• Avisar bnd! → aviam estes tipos alterados  
e DDL alterados ✓

criar Trigger verifica se  
é dia de professor, orientad. ✓

criar trigger para verifica  
se só está ✓



Aluno → Professor

Professor

Professor → Orientador

Aluno: Pedro  
eror trigger na tabela ~~atégo~~, o que se  
deixar só o id depois do atégo terminal -?

SP-Procedure dano ✓

No SQL Server podemos associar os objetos a si mesmos,  
é mais difícil associar o vice versa.

Também possível eror SP's com permisões diferentes  
para determinados usuários UDF e triggers.

Create PROC ...

WITH EXECUTE AS SUPER

↳ permissões iguais à quem criou

CALLER

↳ permissões de quem chama ou

CREATE ASSEMBLY

Isto é útil para o PROC, pode ter instruções que chamam  
a SP n term e assim é invisível para gente

Também é possível cifrar atributos

Aluno: Pedro

### Análise da base de dados

Critérios Informais - clareza dos atributos

- realista se existe ~~entre~~ na  
semântica entre os atributos e a  
relação

Obrigatório → redução de nulls

↳ por vezes temos tipos que não se aplicam  
à relação, deve-se unir com outra tabela.

Dependências formais

$Y \rightarrow Y$

↳ define unicidade Y

Dependência funcional → atributo depende de outro por  
de chave

" Total → atributo que depende da totalidade  
da chave da relação

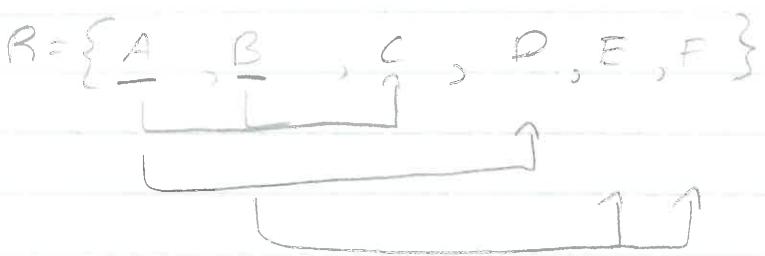
" Transitivo → um atributo que não é chave  
depende de outro e que não é chave e vice-versa.

Normalização → RD Reduzir redundância

1 FN → Atenuabilidade dos atributos, não há obj. ultravalor  
nem nested relations  
↳ pelotomia repetição

2 FN → FD Eliminar dependências parciais

SSN	PhoneNumber	Address	EmployeeName	PlazaLocation
J	J	J	J	J

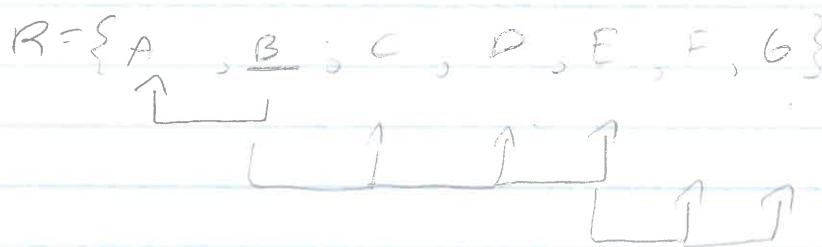


$$R_1 = \{A, C\}$$

$$R_2 = \{B, E, F\}$$

$$R_3 = \{A, B, C\}$$

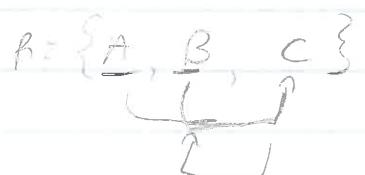
3FN → Elimina as dependências ~~funcionais transitivas~~



$$R_1 = \{A, B, C, D, E\}$$

$$R_2 = \{E, F, G\}$$

BCFN → por vezes todos os atributos dependem do chaves mas parte dessa chave depende de um atributo



$$R_1 = \{B, C\} \text{ perde os DF}$$

$$R_2 = \{A, C\}$$

Deve existir equilíbrio

$$R = \{A, B, C, D, E, F, G, H, I, J\}$$

1FN

$$R_1 = \{A, B, C, D, F\}$$

$$R_2 = \{A, \underline{\underline{G}}, H, I\}$$

2FN

$$R_1 = \{A, B, C, D, F\}$$

$$R_2 = \{A, G\}$$

$$R_3 = \{G, H, I\}$$

3FN

$$R_1 = \{A, B, F\}$$

$$R_2 = \{A, G\}$$

$$R_3 = \{G, H, I\}$$

$$R_4 = \{B, C, D\}$$

do Ano

Single-Level Ordered

Anomônico cada ido do atributo intido

Primário → índice sobre o chão primário

clustered → " " → o chão que é  
é chão primário → conjunto de tipos

Secondary → índice para u. tabela de ~~intidos~~ chaves  
unicas

prior index e nth plus n'ros

Multi-level → índice para um índice de tabelas  
na tentativa de reduzir o número de log<sub>2</sub> bi para 2 log<sub>10</sub> bi

Sug. de indices Compromise entre parceria se  
não houver muitos pesquisas → num determinado  
e simulações nesse caso chão e parceiros se não houver  
muitas modificações nesse mesmo chão

Se a pesquisa é feita por um atributo que é a PK ent.  
então talvez seja mais eficiente criar um índice

clustered

no tabela de PK's aponta para no tabela de  
de índices ou para no tabela onde os ~~todos~~ dados  
associados à opção chão se encontram

Non clustered

A partir de dados que não é os PK são contidos  
tabelas de paginagem com o PK  
associando os dados

Cada índice consiste para n. inst e numero podo  
listagens de outros

caso

caso índice para o nome da alma  $\checkmark$  ~~possivelmente~~

SOL seres operas possui índices clustered e non-clustered

→ podemos alterar o fill factor de maneira a  
a de o page-split → altera com menos frequência,  
evitando assim as operações que são muito engorilhas de  
CPU.

Também deve ser feita o desfragmentação do bloco do  
índice regularmente, para eliminar espacos rotulos que podem  
surgir após operações de reorganização ou page split

ALTER INDEX "no e" OR TOTERATE REORGANIZE

Podemos também alterar o fill factor para algo mais  
adequado à situação

esta desfragmentação

tem em conta o fill factor

Programação em SQL

case

WHEN 1 THEN 'Mora'

WHEN 2 THEN 'sulca'

ELSE + ' ou desconhecida'

FND

visitas nos inner levels  
várias operações dentro  
de um só bloco de tempo

É possível declarar ~~as~~ tabelas temporárias #Table to e tabelas temporárias globais ## é eliminadas quando o último sessão acaba

As tabelas também podem retornar mas têm um scope mais limitado, terminam quando o bloco é fechado, chega ao final ou quando se violam restrições de FK

Consórcios → operações de que são executadas sequencialmente. Fazendo isso permite processar os valores retornados de uma operação SELECT

Estas operações são bastante exigentes para o motor de base de dados e devem ser evitadas

Sintaxe:

```
DECLARE C CURSOR FAST_FORWARD  
FOR SELECT * FROM
```

~~GO~~ @OPEN

OPEN C; → 6 cursor tem de ser aberto

DECAR @x INT

WHILE @@FETCHSTATUS = 0

BEGIN

FETCH ## INTO @x

CLOSE C; → Têm de ser fechado

DELOCATE C → É o o - único bloco

@@error → o 6 valor de saída de cada instrução é retornado neste variável

@@rowcount → nº de linhas afetadas

RAISERROR → Retorna mensagem de erro ao utilizador

RAISERROR ('ERROR msg', severity code, 1, [options])

pode es cifrar os SPs para que não seja possível alterar o conteúdo do SP

CREATE PROCEDURE ...

WITH ENCRYPTION

Mais rotinas dos SPs → São o melhor maneira de que um utilizador tem para executar operações sobre a base de dados sem ter de saber o seu estrutura interno

- A sua execução é eficiente

- Restringindo o acesso do utilizador

o SP, eliminando o risco de

cripulação

UDF's podem ser usados & com fonte de dados, pode-se passar para elas os contrário das views

UDF scalar → Retornam um valor, podem ser usados

com qualquer tipo de em check constraints

Não é permitidas certas operações como

update à base de dados, re-

TRIGGER catch

UDF Table value → retornam tabelas, estendendo de um SELECT

ex:

UDF Multi-Statement Table-valued → É possível combinar o todo de ser possível de combinar código complexo com a capacidade de retornar uma tabela

Sintaxe:

```
CREATE FUNCTION dbo.dbo (@price_order)
RETURNS @Table (no e varchar(100), ssn INT)
```

AS

BEGIN

```
INSERT @Table SELECT *
FROM x
RETURN
```

END

BEGIN

```
INSERT @Table (code, EffectiveDate, Price)
SELECT code, null, org(pPrice)
FROM :
```

RETURN

END

SELECT \* FROM dbo.dba 15,6

Os 3 tipos de UDF's podem ser definidos com Schema Binding, o que ~~permite~~ adiciona informação a um o tabela mas não pode alterar ou eliminar → vantagem sobre SP

```
CREATE FUNCTION dbo.x (...) RETURNS ...
WITH Schema Binding
```

Trigger

Se-chamts o SP mas é ativado por certos eventos  
e suportados em DDL e DML

Comprissão entre integridade e desempenho

No DML há dois tipos de triggers

Instead of → Verificamos algumas condições e  
efetua | depois chama-se normalmente a operação,  
| caso o trigger é recursivo n  
rois ser chamado novamente  
po de ossuir  
que todos os dados já possuem constraints

```
CREATE TRIGGER nae ON Tablex AFTER Instead of
```

INSERT, DELETE, UPDATE

AS

pode os criar ou desativar triggers

```
ENABLE / DISABLE TRIGGER nae ON Tablex
```

Com os triggers te-los associado à tabela temporária insert onde se encontram os dados que irão ser inseridos na tabela.

AFTER → pode os ter rômos por todo

INSTEAD OF → Apenas naquela

Com TRIGGERS temos acesso a ~~as tabelas~~  
tabelas lógicas Deletd e INSERTED

Bi

INSERT → inserted

DELETE → DELETED

UPDATE → DELETED e INSERTED

J

Tem scope muito limitado

Nos triggers ainda temos acesso a ~~as tabelas~~  
que nos permitem sobre avisar as alterações  
TF UPDATE (Column)

columns-updated → retorna bit por bit op

L, U, D operações → não é permitido  
CREATE, ALTER, DROP,  
etc

Transação → conjunto de operações de Escrita e Leitura  
da base de dados  
intuitivo

Por default ~~for~~ todos os comandos SQL é transação

Propriedades de uma transação

Atomicidade → Se a operação é completa ou não é feita

Integridade → Após todos as operações um estado de integridade  
tem que se manter.

Isolamento → Cada transação deve ser executado de  
modo único e o ~~deve ser~~ ser processado

o recurso deve ser feito em exclusivo

Presistência: Os efeitos de uma transação são visíveis  
para as outras transações, todos anteriores não devem afetar  
operações futuras

• Isolamento → Recurso da técnica de scheduling que  
definem a ordem pela qual é feitas as operações

Escalonamento Serializado - a transação é executada de  
modo a vez, pouco eficiente

Concorrente Serializado - As transações são executadas  
em simultâneo desde que não existe interferência na  
execução de ambas. Ex: Enquanto operações via I/O  
for operações de I/O lentas, outra pode executar outras operações  
mais eficiente

O escalonamento concorrente pode levar a vários problemas de  
como é essa das leituras sujas, ou atualizações perdidas.  
Isto deve-se ao fato de operações distintas ~~não~~ podem usar  
do mesmo recurso.

~~Outra~~ operação perdida → a transação escreve  
informações que já não são lida e escrita por outra

Leitura suja → a transação só é um valor, que  
devido a algum sínolo de uso foi alterado por  
outra operação de rollback, quando o seu valor é  
consequentemente a um inconsistência de dados por parte da  
transação transação que o alterou antes do rollback

## Métodos de controlo de concorrêncio

Mecanismos de Locking - Quando a transacção

início é-lhe atribuída uma etiqueta sequencial quando  
accede a um elemento recorde, se outra transacção aceder  
e o não é sup. do seu respectivo recorde é bloqueado

Mecanismos de Locking - uma transacção só pode  
acceder a um elemento se esse elemento estiver desbloqueado, quando  
o tempo no seu passo esse elemento é bloqueado, o poro  
além de locks de acesso também há locks de leitura  
e escrita. → obriga a implementação de politicas de  
prevenção de deadlock

Ainda há métodos alternativos que consideram que  
existem intersecções entre os transacções

## Recuperação de faltas

Todos os sistemas estão suscitos a faltas, e → O SGBD  
não é exceção e por isso deve estar preparado

## Garantida

- - graves → faltas nas transacções → necessários de Escalonamento
- + graves → Pode parcial ou total de dados → Backups

Backup

## Transaction log

Com as técnicas de recuperacão por escalonamento  
temos recuperadores, e não recuperáveis

Com objetos com casco e ~~sem~~ objetos em casco  
Escritas e não escritas

Recuperável e não recuperável → um escalonamento é recuperável  
se nenhuma trans. for concluída de que todos os outros  
que operaram no mesmo também tiveram sido concluídos

## Recuperáveis

Com e sem objetos em casco → objetos com casco devem  
ser evitados. → tornam o processo de recuperação pouco  
eficiente

Para evitar de uma transacção só pode ler um  
objeto, ~~negar~~ de que está a ser usado por outra, se está  
só a ler para o seu execução

Recuperável, com objetos em casco, escrita → se no centro  
transacção só escreve num elemento após outra transacção que  
estava a usar esse mesmo elemento só feito ~~também~~ tiveram  
de ser concluídos os seus execuções.

## Backups

Cópias de segurança efectuadas com regularidade, só permite  
recuperar a até determinado ponto.

Como as operações de backup são exigentes para o sistema, queremos  
que sejam feitas com concorrência e que o SGBD esteja  
a ser pouco usado.

Transaction logs → registram todas as operações feitas na  
BD, incluindo commits.

Qtz Por questões de eficiência de I/O o dado é escrito  
em buffers e só depois em disco

transaction logs reporta-se pelo disco e para reconhecimento, os registos das operações têm de ser feitas a ser escritos a disco antes do registo do commit. Também guardam uma imagem dos dados antes e depois das alterações.

Em caso de falha os backup fazem com as transaction log. Em caso de falha fazem o roll forward reconstruindo a base dados com o backup e de seguida fazendo as transaction logs com os after e before images.

Em caso de falhas nas transacções basta usar o before image capturado antes da transacção.

Em caso de falha do sistema operativo ou do SGBD, também se faz o rollback até ao último momento em que a base dados deixou de ser intage.



Ate' que ponto?

• um simples rollback não obriga o troço de dados a fazer rollback até ao ponto de backup, pode estar afastado. Existem mecanismos de savepoint que registam até que ponto se deve fazer o rollback.

Consequências das regras de recuperação de falhas

- necessidade de espaço no disco
- mais lidos e escritos
- mais uso de CPU

### Aspectos de Segurança

Para ter login para ter acesso a uma BD tem de estar associado a um userID, pode um login estar associado a muitos userID.